Sublinear Secure Computation from New Assumptions

Elette Boyle[1], Geoffroy Couteau[2], and Pierre Meyer[3]

[1] Reichman University and NTT Research. eboyle@alum.mit.edu
[2] Université Paris Cité, IRIF, CNRS. couteau@irif.fr
[3] Reichman University and Université Paris Cité, IRIF, CNRS.
pierre.meyer@irif.fr

Abstract. Secure computation enables mutually distrusting parties to jointly compute a function on their secret inputs, while revealing nothing beyond the function output. A long-running challenge is understanding the required communication complexity of such protocols—in particular, when communication can be *sublinear* in the circuit representation size of the desired function. For certain functions, such as Private Information Retrieval (PIR), this question extends to even sublinearity in the input size.

We develop new techniques expanding the set of computational assumptions for sublinear communication in both settings:

- **Circuit size.** We present sublinear-communication protocols for secure evaluation of general layered circuits, given any 2-round rate-1 batch oblivious transfer (OT) protocol with a particular "decomposability" property. In particular, this condition can be shown to hold for the recent batch OT protocols of (Brakerski et al. Eurocrypt 2022), in turn yielding a new sublinear secure computation feasibility: from Quadratic Residuosity (QR) together with polynomial-noise-rate Learning Parity with Noise (LPN).
  Our approach constitutes a departure from existing paths toward sublinear secure computation, all based on fully homomorphic encryption or homomorphic secret sharing.
- **Input size.** We construct single-server PIR based on the Computational Diffie-Hellman (CDH) assumption, with *polylogarithmic* communication in the database input size n. Previous constructions from CDH required communication $\Omega(n)$. In hindsight, our construction comprises of a relatively simple combination of existing tools from the literature.

Keywords: Foundations · Private Information Retrieval · Secure Multiparty Computation

1 Introduction

Secure computation enables mutually distrusting parties to jointly compute a function on their secret inputs, while revealing nothing beyond the function

output. We focus on the case of two-party computation with semi-honest (passive) security. Since the seminal feasibility results of the 1980s [Yao86, GMW87, BGW88, CCD88], a major challenge in the area of secure computation has been if and when it is possible to break the "circuit-size barrier." This barrier refers to the fact that all classical techniques for secure computation required a larger amount of communication than the size of a boolean circuit representing the function to be computed. In contrast, insecure computation only requires exchanging the inputs, which are usually considerably smaller than the entire circuit.

Early positive results with sublinear communication either required exponential computation [BFKR91, NN01], or (as discussed later) were limited to very simple functions such as point functions [CGKS95, KO97, CG97] or constant-depth circuits [BI05].

Beyond the circuit-size barrier. This situation changed with the breakthrough result of Gentry [Gen09] on *fully homomorphic encryption* (FHE). FHE is a powerful primitive supporting computation on encrypted data, which can be used to build optimal-communication protocols in the computational setting [DFH12, AJL+12], by having parties perform the desired computation *locally* on encrypted inputs without additional communication. However, despite significant efforts, the set of assumptions under which we know how to build FHE is very narrow. Standard approaches are restricted to lattice-based assumptions, such as Learning With Errors (LWE), and in particular do not include any of the traditional assumptions which were used in the 20th century. Very recent developments in indistinguishability obfuscation imply results based on an alternative (relatively exotic) bundle of assumptions [CLTV15, JLS22].[4]

The work of [BGI16] first showed that secure computation with communication sublinear in the circuit size could also be based on assumptions not known to imply FHE, via a new primitive of *homomorphic secret sharing* (HSS). HSS can be viewed as a relaxation of FHE, where homomorphic evaluation can be distributed among two parties who do not interact with each other. More concretely, from the Decisional Diffie-Hellman (DDH) assumption, [BGI16] constructed a form of HSS for branching programs (including NC^1), implying secure computation for the corresponding function class with asymptotically optimal communication. In turn, this was shown to yield secure computation for general layered circuits[5] of size s with sublinear communication $O(s/\log s)$, by evaluating in $(\log s)$-depth blocks, and communicating only between blocks.

Since then, the HSS-based approach and variations have resulted in sublinear-communication secure protocols from an additional assortment of assumptions. Following the [BGI16] blueprint, the works of [FGJI17, OSY21, RS21] were able to replace the DDH assumption with Decision Composite Residuosity (DCR). The

[4] Namely, subexponential security of the combination of: Learning Parity with Noise, plus polynomial-stretch pseudorandom generators in NC^0, plus the Decision Linear assumption on symmetric bilinear groups of prime order [JLS22].

[5] A depth-d circuit is layered if it can be divided into d layers such that any wire connects adjacent layers.

framework was recently abstracted and extended to further algebraic structures, including a class of assumptions based on class groups of imaginary quadratic fields [ADOS22]. In addition, the work of [CM21] built HSS for $\log \log$-depth circuits (yielding $O(s/\log \log s)$ communication secure computation for layered circuits) based on a strong flavor of the Learning Parity with Noise (LPN) assumption: with a small number of samples, but assuming super-polynomial hardness, with inverse-superpolynomial noise rate.

To date, these two approaches—FHE and HSS—still comprise the only known paths to sublinear-communication secure computation for general circuit classes, without resorting to superpolynomial computation or setup assumptions such as correlated randomness [IKM+13, DNNR17, Cou19]. It remains a motivated research agenda not only to continue expanding the set of distinct computational assumptions upon which sublinear secure computation can be built, but additionally of exploring new types of approaches toward this goal.

Private Information Retrieval. As mentioned, one exception to the above treatment is the special case of specific simple functionalities: most prominently, the task of Private Information Retrieval (PIR) [CGKS95, KO97]. A (single-server) PIR protocol roughly amounts to a secure computation protocol (with one-sided privacy) for the specific function $f(x, i) = x_i$ with $x \in \{0,1\}^n$ and $i \in [n]$. Unlike the case of general computation (where the communication complexity of the underlying function may be $\Omega(n)$ even without security), PIR can admit secure protocols with communication *sublinear (even polylogarithmic) in the input size.*

For many years, protocols for PIR with polylogarithmic communication in n were known only from the Decisional Composite Residuosity (DCR), Learning with Errors (LWE), or Phi-hiding assumptions [CMS99, Cha04, Lip05, OS07]. More recently, such constructions were achieved from Quadratic Residuosity (QR), or Decisional Diffie-Hellman (DDH) [DGI+19].

1.1 Our Results

We present new approaches and techniques for both of the above settings, ultimately extending the set of computational assumptions under which we can achieve sublinear-communication secure computation protocols.

Our results fall within two primary categories:

- We obtain (slightly) sublinear secure two-party computation for general layered circuits, through a new path of low-communication batch oblivious transfer.
- We explore the specific goal of Private Information Retrieval (PIR), and provide a new construction with polylogarithmic communication based on *Computational* Diffie-Hellman (CDH).

We emphasize that our protocols execute in polynomial runtime, and do not rely on any correlated randomness assumptions.

Sublinear 2PC for layered circuits. We present a new approach toward secure two-party computation protocols for general layered circuits, with communication complexity that scales sublinearly in the circuit size. As opposed to building FHE or HSS, our approach begins with protocols for "batch Oblivious Transfer" with low communication.

Oblivious Transfer (OT) is an atomic functionality in which sender and receiver parties begin with inputs $m_0, m_1 \in \{0,1\}$ and $b \in \{0,1\}$, respectively; at the conclusion the receiver learns the selected message m_b; and neither party learns further information about one another's inputs. OT was shown to be a complete functionality for general secure computation [Kil00], where OT protocol execution(s) take place for each nonlinear gate of the corresponding circuit.

OT protocols are known from a number of standard assumptions, in just two rounds of communication (i.e., one message from receiver to sender, and one message in return); but, the communication complexity for all such solutions is (inherently) significantly larger than the input size. Very recently, it was shown by Brakerski et al. [BBDP22] how to achieve a *batched* version of OT, still in two rounds, and with *rate-1* communication. That is, for a collection of message pairs $(\{m_0^{(i)}, m_1^{(i)}\})_{i\in[k]}$ and selection bits $(b^{(i)})_{i\in[k]}$, a sender and receiver could perform k parallel batched executions of OT in communication roughly k.

We prove that any such protocol which satisfies an additional *decomposability* property suffices to imply secure computation protocols for general layered circuits with sublinear communication complexity. To define decomposability, consider the communication structure of any 2-round rate-1 batch OT protocol. In the first round, the receiver sends $k + o(k)$ bits to the sender,[6] somehow encoding its selection bits $b^{(i)}$. In response, the sender performs some computation as a function of its message pairs $\{m_0^{(i)}, m_1^{(i)}\}$, and returns $k + o(k)$ bits in response, somehow encoding the k selected messages, $m_{b^{(i)}}^{(i)}$. For the constructions of [BBDP22], the sender's message size is just $k + \mathsf{polylog}(k)$.

We say that the (2-round, rate-1) batch OT protocol is *decomposable* if for any agreed subset $S \subset [k]$ of indices, the sender can choose a corresponding subset of $|S| + \mathsf{polylog}(k)$ of its return message bits, such that sending this partial sender response reveals *exactly* the corresponding subset of selected messages $(m_{b^{(i)}}^{(i)})_{i\in S}$ to the receiver. Namely, given the partial response, these $|S|$ messages can be recovered, and no information is revealed about $m_{b^{(i)}}^{(i)}$ for $i \notin S$.

Theorem 1 (Sublinear 2PC from Decomposable Batch OT - informal). *Assume existence of 2-round rate-1 batch OT with the above "decomposability" property. Then for any k, we can securely compute layered (synchronous) circuits of depth d and size s using $\mathsf{poly}(2^{2^k}, s)$ computation and $O(2^{2^k} \cdot d \cdot \mathsf{poly}(\lambda) + s/k)$ communication.*

In particular, for $k = O(\log\log s)$, we obtain communication $O(s/\log\log s + d^{1/3} \cdot s^{2(1+\varepsilon)/3} \cdot \mathsf{poly}(\lambda))$, for an arbitrary small constant ε. The latter is sublinear in s whenever $d = o(s^{1-\varepsilon}/\mathsf{poly}(\lambda))$, i.e.*, the circuit is not too "tall and skinny".*

[6] Our construction can actually handle arbitrary *constant* client-to-server upload rate, as long as the sender-to-receiver download rate is 1.

This decomposability property is not simply hypothetical, but rather was inspired by the batch-OT protocols of Brakerski et al. [BBDP22], which we show to satisfy the requirement. At a high level, the sender's message in their protocols consists of an encryption of the selected message bits (computed homomorphically as a function of receiver-sent ciphertexts of its selection bits, together with the message pairs $\{m_0^{(i)}, m_1^{(i)}\}$), compressed à la [DGI+19] to rate 1. The resulting rate-1 ciphertexts have the structure of a $\mathsf{polylog}(k)$-size "header" string, independent of the messages, together with a *single bit* of information for each encrypted message bit. Decomposability thus follows (pseudo)directly, by simply omitting those information bits corresponding to encrypted messages the sender wishes to drop (i.e., $[k] \setminus S$).[7]

In turn, we obtain the following corollary.

Corollary 2 (Sublinear 2PC from QR+LPN - informal). *The conclusion of Theorem 1 holds based on Quadratic Residuosity (QR) and Learning Parity with Noise (LPN) for any inverse-polynomial noise rate.*

Our result is summarized on Table 1, where we also recall the state of the art in sublinear secure computation. We remark that while sublinear $O(s/\log\log s)$-communication protocols were known from a variant of LPN from [CM21], their result must assume superpolynomial hardness of LPN with a small inverse-superpolynomial error rate. In contrast, our result requires only polynomial hardness of LPN, with any inverse-polynomial error rate (as inherited by the construction of [BBDP22]).

We finally mention that this result is also not implied by the constructions of pseudorandom correlation functions (PCF) [BCG+20] from QR+LPN of [OSY21] (or in fact any of the line of work on pseudorandom correlation generators (PCG) [BCG+19]). While PCG/PCFs enable the generation of large quantities of *random* instances of OT with sublinear communication, the best known approaches for utilizing these random correlations within an actual secure computation protocol require communication that scales linearly with the circuit size.

Private Information Retrieval. Motivated by the goal of building decomposable rate-1 batch OT from new assumptions, we then turn to a deeper exploration of one of the required underlying components from the [BBDP22] batch OT construction: (single-server) Private Information Retrieval (PIR).

We succeed in constructing PIR with polylogarithmic communication from the *Computational Diffie-Hellman* assumption. While this is only one sub-component required to obtain the necessary batch OT from LPN+CDH,[8] this provides one step toward this direction. But, more importantly, it constitutes a new feasibility result of its own right. From CDH, previously no PIR protocol was known with communication $o(n)$.

[7] We are of course sweeping details under the rug here, and refer the reader to the main body for a more complete treatment.

[8] Indeed, the approach of [BBDP22] requires also a form of homomorphic encryption compressible to rate 1.

Table 1: Existing protocols for secure computation with sublinear communication under various assumptions, in the computational setting.

	Assumptions	Circuit class	Sublinearity[1]
[Gen09]	LWE	$\mathsf{P/poly}$	$O(n+m)$
[BGI16]	DDH	Layered circuits	$O(n+m+s/\log s)$
[OSY21, RS21]	DCR	Layered circuits	$O(n+m+s/\log s)$
[CM21]	superpoly LPN[2]	Layered circuits	$O(n+m+s/\log\log s)$
[ADOS22]	Class groups	Layered circuits	$O(n+m+s/\log s)$
This work	LPN + QR[3]	Layered circuits	$O\left(n+m+d^{1/3}\cdot s^{2(1+\varepsilon)/3}\cdot\mathsf{poly}(\lambda)+\frac{s}{\log\log s}\right)$

[1] We use n for input size, m for output size, s for circuit size, and d for circuit depth.
[2] [CM21] assumes the superpolynomial hardness of the LPN assumption with dimension N, $O(N)$ samples, and noise rate $N^{o(1)-1}$.
[3] We assume the polynomial hardness of LPN with dimension N, $\mathsf{poly}(N)$ samples, and inverse-polynomial noise rate.

Theorem 3 (PIR from CDH - informal). *Based on the Computational Diffie-Hellman (CDH) assumption, there exists single-server PIR on n-bit databases with communication $\mathsf{polylog}(n)$ and $\mathcal{O}(\log(n))$ rounds.*

In hindsight, our construction forms a surprisingly simple and clean combination of two existing tools from the literature. Along the way, we identify an improved procedure for converting between a weak form of "semi-PIR" as considered in [BIP18], which reveals the client's queried index with some probability, to full-blown secure PIR. We refer the reader to the Technical Overview for more details.

2 Technical Overview

We assume familiarity with standard cryptographic assumptions such as QR, LPN, CDH, and DDH, and refer the reader to the full version for a formal statement of these assumptions.

2.1 Sublinear 2PC for Layered Circuits from Decomposable Batch OT

We consider Boolean circuits over any base of gates with fan-in two.

Toward our sublinear 2PC result for layered circuits, we begin by focusing on circuits of low depth k (e.g., think of $k = \log\log\log s$), and devise a secure protocol with communication $n + m + (2^{2^k} \cdot \mathsf{poly}(\lambda))$, for input size n, output size m, circuit size s, and security parameter λ. Given such a tool, we can appropriately divide a larger layered circuit into depth-k blocks where the sum of all block input and output sizes is s/k, and then iteratively compute (secret shares of) each layer output via the sub-protocol. Combined, this yields a secure computation for the layered circuit with overall communication $O(s/k + 2^{2^k} \cdot d \cdot \mathsf{poly}(\lambda))$, as desired.

Starting point: An SPIR viewpoint. Consider a circuit with input size n, output size m, and low depth k. Given fan-in 2, each output bit is computed as a function of at most 2^k input bits. We may thus view the circuit output as dictated by m separate truth tables, each of size 2^{2^k}, indexed by the values of the corresponding relevant 2^k input bits. More concretely, think of one party as holding the (partially collapsed) truth tables incorporating its known inputs, and the second party as holding its own input string, dictating the relevant position of each truth table. We will refer to the first party as "sender" and second as "receiver."

Given this perspective, protocols for (Symmetric) Private Information Retrieval (SPIR) immediately come to mind. An SPIR protocol is a strengthened version of PIR, where the client additionally learns nothing beyond its queried value of the database. Secure computation of our circuit precisely amounts to m instances of SPIR, where the receiver party learns exactly the desired indexed values of the m truth tables.

However, the situation is not so simple: Even the best known (S)PIR protocols have communication polylogarithmic in the database size. Applying m instances of SPIR for the m outputs would thus yield communication $\mathsf{polylog}(2^k) \cdot m \in \Omega(km)$, killing sublinearity.

In order to obtain sublinear communication, we must somehow leverage that the m SPIR instances are not completely independent, but rather are made with *correlated* queries. That is, although there are m instances each with (2^k)-bit index values, the $m \cdot 2^k$ selection bits have several repeats, collectively coming from different subsets of only $n < m \cdot 2^k$ input bits.

Toward batch SPIR with correlated queries. Our task becomes precisely to construct such an object: m-instance batch SPIR, with significantly lower communication complexity given correlated queries.

For purposes of discussion, suppose there existed a 2-round rate-1 protocol for oblivious transfer, where each sender and receiver (magically) sends only a single bit. Given access to such a tool, then by leveraging ideas from the literature (e.g., achieving PIR from linearly homomorphic encryption [KO97]), we would be set. Indeed, the receiver would simply send 1 bit for each input bit, corresponding to the first OT message using this value as a selector bit. These first messages could then be *reused* by the sender in multiple, recursive executions.

More concretely, suppose the server holds a database of N bits and that the receiver wants to retrieve the element stored at index $x = (x_1, \ldots, x_{\log N})$. If the receiver sends a message otr_1 generated as its first-round OT-receiver message for the first bit x_1 of the desired index, the server can take the database, pair up elements whose indices differ only on the first bit, then apply the OT-server computation with respect to otr_1 on each pair in order to retrieve a single-bit response for each, creating a new "database" of half the number of elements, each corresponding to a 1-bit sender answer message. If instead the receiver sends messages $(\mathsf{otr}_1, \ldots, \mathsf{otr}_{\log N})$, one for each bit of the desired index, the server can now iteratively compress the database down to a single bit by building a "Merkle tree" where in each recursive iteration corresponding to input index

bit x_i, the new "database" is split into pairs of messages whose indices differ only in this index, and performing the OT-server computation on each pair produces a new list of 1-bit sender answer messages of again half the length. At the conclusion, the server will be left with a single message value remaining, which by construction precisely enables the receiver to recover the target value stored at index x. This approach extends directly for m distinct databases with the *same* total receiver message $(\mathsf{otr}_1, \ldots, \mathsf{otr}_{\log N})$, since the corresponding OT-receiver messages can be used independently in any mix and match format across databases. In turn, the sender would need to send only m total bits response, one bit for each database query.

Of course, unfortunately, we do not have such a strong rate-1 OT. We thus turn to the next closest alternative which does exist: 2-round rate-1 *batch* OT, as recently achieved by Brakerski et al. [BBDP22]. Batch OT considers a collection of ℓ message pairs $(\{m_0^{(i)}, m_1^{(i)}\})_{i\in[\ell]}$ and selection bits $(b^{(i)})_{i\in[\ell]}$, and enables a sender and receiver to perform ℓ parallel batched executions of OT with communication roughly ℓ. Attempting to apply the above strategy with rate-1 batch OT, however, poses significant challenges.

- The batching structure restricts the "mix and match" abilities of the sender when using the receiver's OT message. The sender must respond to the *entire* batched vector of receiver's selection bits at any stage, without freely accessing subsets of selection bits. Instead, the above approach involves using each selection bit $b^{(i)}$ within a *different* number $(N/2^i)$ of message pairs.
- Even worse, the sender's (batch) response in general is only defined given *all ℓ pairs of messages* to be selected by the bits $b^{(1)}, \ldots, b^{(\ell)}$. In contrast, the above approach crucially relies on the ability to choose the message pairs for selection bit $b^{(i)}$ *dynamically* as a function of the server's responses given the previous selection bits $b^{(1)}, \ldots, b^{(i-1)}$.
- Finally, it is no longer the case that for each selected message the sender has a *single* corresponding response bit. In fact, rate 1 here does not even mean that for ℓ instances that exactly ℓ bits are sent in each direction, but rather just asymptotically $\ell + o(\ell)$. This means that in each recursive OT execution, the sender's messages (and thus "database entry" size) may grow, leading to large growth and ultimately large communication upon further recursions.

Decomposable batch OT. With this motivation, we introduce the notion of *decomposable (2-round, rate-1) batch oblivious transfer*, which can be seen as a strengthening of two-round batch OT with constant upload-rate (*i.e.* the size of the receiver message is linear in the batch size ℓ) and download-rate asymptotically one (*i.e.* the size of the sender message is $\ell + o(\ell)$). The differences boil down to a notion of *decomposability* which we impose on the sender message.

At a high level, what we want to capture is the fact that the receiver should be able to retrieve the i^{th} selected message in the batch if and only it also has access to the i^{th} bit of the sender message (using its own internal state saved from generating the receiver message). More generally, given only a subset of the bits of the sender message, the receiver should able to retrieve the corresponding subset of selected messages in the batch.

Slightly more formally, we say that the (2-round, rate-1) batch OT protocol is *decomposable* if for any agreed subset $S \subset [\ell]$ of indices, the sender can choose a corresponding subset of $|S| + \mathsf{polylog}(\ell)$ of its return message bits, such that sending this partial sender response reveals *exactly* the corresponding subset of selected messages $(m^{(i)}_{b^{(i)}})_{i \in S}$ to the receiver. Namely, given the partial response, these $|S|$ messages can be recovered, and no information is revealed about $m^{(i)}_{b^{(i)}}$ for $i \notin S$.

For our purposes, it will suffice to consider a relaxation of the notion we just described, and allow the sender message to have some small overhead rather than having a one-to-one correspondence between the bits on the sender message and the ℓ selected messages. In this relaxed form, we require that the sender message be comprised of two parts: a "reusable" part (of size $o(\ell)$), and a "decomposable" part (of size ℓ). On its own, the reusable part should reveal nothing about the messages, but can be used to "decode" each bit of the decomposable part so as to retrieve (exactly) the corresponding selected message in the batch. Among other benefits of this relaxation, it allows us to consider constructions whose download-rate is only asymptotically one.

This decomposability property is not only enough for our needs, but perhaps more importantly, is *achievable*, in fact achieved by the batch OT constructions of [BBDP22]. Roughly speaking, the sender message in their construction is composed of a rate-1 encryption of the vector of requested message bits, with structure consisting of a short "header" independent of the message bits, together with a single ciphertext bit encoding each message bit separately. Decomposability can then be achieved by sending only those ciphertext bits encoding the desired subset of messages.

Slightly more accurately, this describes the situation for all but an inverse-polynomial fraction of message bits (corresponding to noisy coordinates of an LPN ciphertext sent by the receiver), which actually encode the *incorrect* messages. In order to separately address these values, they employ a "co-PIR" (or "punctured OT" [BGI17]) to efficiently mask out the undesired values from the receiver, and a separate PIR to learn the correct values for these positions. The separate PIR query responses appear as part of the short "header" information of the server's response, which may sound like an issue, as this portion should not reveal information directly about any message bits. However, this problem does not occur, because the extra PIR queries are set up to actually reveal the *difference* between the masked-out incorrect message $(r_i \oplus m_{1-b})$ and the target message m_b. Because of the mask, this difference value (revealed in the header) provides no information about any message in the absence of the corresponding value $(r_i \oplus m_{1-b})$ from the payload portion of the ciphertext, as required by decomposability. We refer the reader to Section 3.2 for further details.

Sublinear 2PC from decomposable batch OT. This decomposability property directly allows us to address one of the above challenges of batch OT: we will not have issues with exponential growth of the database entry size in the recursive OT executions. Instead, the result of one iteration of the batch OT on n inputs

will result in a short $o(n)$-size header together with n bits that each provide information about a distinct queried message. The header string we will put to the side (ultimately we will send the collection of all the headers, which is still sufficiently short). The remaining n bits induce the recursive sender-message database that, as desired, consists of exactly 1 bit per message.

In fact, if we temporarily suppose that the assignment graph structure of n input bits to $m = n$ output bits can be decomposed as the disjoint union of 2^k matchings, then we have a solution. Each disjoint matching will correspond directly to a different instance of n-input batch OT, where each of the n inputs is simultaneously used to index a different database. Applying the recursive solution as above, the sender will ultimately compute a single bit for each output, as well as a collection of header strings from each of the batch OT executions.

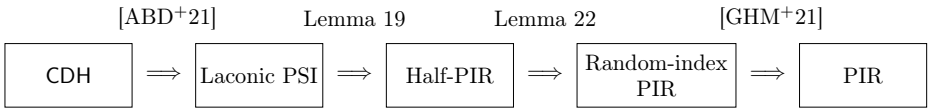
The remaining challenge is that general circuits do *not* have such nice regular structure, instead with inputs appearing in different numbers of output computations, with inconvenient correlations, demanding a stronger form of "mix and match" of batched OT queries beyond a direct approach.

To address this issue, we modify the structure of batch OT receiver queries, effectively extending the batch size (say from n to $2n$), and employing a careful choice of how to pack extra copies of more highly influential input bits into the queried vector, so that the overall total number of batch OT instances remains sufficiently small that the overhead of extra header strings does not negatively impact the final communication complexity. We refer the reader to the technical body for a detailed treatment of this procedure.

2.2 Polylogarithmic PIR from CDH

We now turn our attention to our second contribution: private information retrieval with polylogarithmic communication from the computational Diffie-Hellman assumption. A private information retrieval (PIR) is a two party protocol between a server S holding a string z (the database) and a client C holding an integer i. At the end of the interaction, the client should learn z_i, without revealing i to the server. A polylogarithmic PIR is a PIR where the total communication is $\mathsf{poly}(\lambda, \log |z|)$, where λ is the security parameter.

Below, we sketch our approach to building polylogarithmic PIR from CDH. In hindsight, our construction is in fact relatively straightforward, and follows from an elegant combination of two recent results. We outline the sequence of implications below.



Laconic PSI. A private set intersection protocol is a two-party protocol allowing a receiver to securely compute the intersection of its input set S_R with the

set S_S of a sender: at the end of the protocol, the receiver learns $S_R \cap S_S$ and nothing more. A *laconic* PSI protocol, introduced in [ABD+21], additionally enforces that the protocol is two-round (receiver to sender, then sender to receiver), and both the total communication and the sender runtime are bounded by $\mathsf{poly}(\lambda, \log |S_R|, |S_S|)$. The work of [ABD+21] showed that laconic PSI can be constructed from anonymous hash encryption, a primitive that can be constructed (in particular) from the CDH assumption [DG17b, DG17a, BLSV18].

From Laconic PSI to Half-PIR. Given a laconic PSI protocol, we exhibit a construction of polylogarithmic-communication PIR, using in addition a pseudorandom function. However, our construction only achieves a very weak form of security: it only guarantees that the index i is kept hidden from the server with probability $1/2$. This notion, which we call half-PIR, has been introduced in [BIP18] (under the name $\mathsf{Rand}\frac{1}{2}\mathsf{PIR}$). It was shown in [BIP18] that polylogarithmic half-PIR already suffices to construct *slightly sublinear* PIR (with communication $O(|z|/\log |z|)$); looking ahead, we will provide a stronger reduction and show that it actually implies polylogarithmic PIR.

Our construction of half-PIR proceeds as follows: the client and the server agree on a PRF key K. The server with input z builds the set $S_R = \{F_K(1||z_1), \cdots, F_K(|z|||z_{|z|})\}$, and the client with input i builds the set $S_S = \{F_K(i||b)\}$, where b is a uniformly random bit. The core properties that this achieves are:

- If $b = z_i$, then $|S_R \cap S_S| = 1$ (note that $|S_S| = 1$), and
- If $b \neq z_i$, then $|S_R \cap S_S| = 0$ with high probability.

To show the second property, we rely on the security of the PRF to argue that a collision between PRF evaluations on distinct inputs is highly unlikely (provided the PRF outputs are large enough). Note, therefore, that we rely on the PRF *security* to argue the *correctness* of the construction (while this is slightly unusual, this kind of arguments has been used a few times in the literature).

Now, the server and the client execute a laconic PSI, which has total communication $\mathsf{poly}(\lambda, \log |z|)$ (since $|S_S| = 1$). At the end of the protocol, the server, who plays the role of the receiver, sends $|S_R \cap S_S|$ to the client. Note that $|S_R \cap S_S| = (1 - b) \oplus z_i$, hence the client can decode z_i from this information. Yet, whenever $|S_R \cap S_S| = 0$, the security of the laconic PSI implies that the server actually learns nothing about i: this guarantees client security with probability $1/2$. When $|S_R \cap S_S| = 1$, however, the server learns the intersection $S_R \cap S_S = F_K(i||z_i)$, and can in particular retrieve i easily.

From Half-PIR to PIR. We now turn to constructing a polylogarithmic PIR from a polylogarithmic half-PIR. Here, our construction is mostly a simple observation: half-PIR implies random-index PIR via a straightforward construction. A random-index PIR, introduced in [GHM+21], is a PIR protocol where the client has no input, and receives (i, z_i) where the index i is picked uniformly at random between 1 and $|z|$. Given a half-PIR, building a random-index PIR is almost immediate: the client and the server execute λ parallel instances of a half-PIR protocol, where the client uses uniformly random independent indices in each

instance. With overwhelming probability, at least one of these instances will be secure (in the sense that the server does not learn the index); the client simply outputs (i^*, z_{i^*}) where i^* is the index used in the first such execution.

Eventually, random-index PIR was recently shown in [GHM+21] to imply full-fledged PIR, with a $\log|z|$ blowup in communication and round complexity. The key observation underlying this reduction is that a single invocation of a random-index PIR, together with sending $\log|z|$ bits, allows to reduce the task of executing a PIR on a size-$|z|$ database to that of executing a PIR on a size-$|z|/2$ database. The construction follows by recursively invoking this construction (we provide a more detailed description of this construction in Section 4.3). Combining all these building blocks together leads to a logarithmic-round, polylogarithmic-communication PIR from the CDH assumption.

3 Sublinear Computation for log log-Depth Circuits

3.1 Decomposable Two-Round Batch Oblivious Transfer

We introduce the notion of *decomposable two-round batch oblivious transfer* (definition 4), which can be seen as a strengthening of two-round batch OT with constant upload-rate and download-rate asymptotically one. The differences boil down to a notion of *decomposability* which we impose on the sender message, which should be comprised of a (small) *reusable part* and a linear-size *decomposable part*: the receiver should be able to retrieve the i^{th} selected message in the batch if and only it also has access to the i^{th} bit of the decomposable part of the sender message (along with the reusable part).

Definition 4 (Decomposable Two-Round Batch Oblivious Transfer). *Let $k \in \mathbb{N}^\star$ and $\alpha(\cdot) = o(n)$. A semi-honest two-round decomposable batch OT protocol with $\alpha(\cdot)$-overhead between a sender and a receiver is defined as a triple of PPT algorithms $\mathsf{dec\text{-}OT} = (\mathsf{dec\text{-}OTR}, \mathsf{dec\text{-}OTS}, \mathsf{dec\text{-}OTD})$ with the following syntax and properties:*

- **Syntax.**
  - $\mathsf{dec\text{-}OTR}$: *On input the security parameter 1^λ and a vector of selection bits $\vec{b} = (b_1, \ldots, b_k) \in \{0,1\}^k$, $\mathsf{dec\text{-}OTR}$ outputs a receiver message $\mathsf{otr} \in \{0,1\}^{\mathcal{O}(k)}$ and an internal state st; without loss of generality we assume that st contains all the random coins used by $\mathsf{dec\text{-}OTR}$ as well as $\vec{b}$.*
  - $\mathsf{dec\text{-}OTS}$: *On input the security parameter 1^λ, a receiver message otr, and a database $((m_0^{(i)}, m_1^{(i)}))_{i\in[k]} \in \{0,1\}^{2k}$ comprised of k pairs of bits, $\mathsf{dec\text{-}OTS}$ outputs a sender message $\mathsf{ots} = (\mathsf{ots}^\star, \mathsf{ots}^{\mathsf{dec}})$, which is comprised of a reusable part $\mathsf{ots}^\star \in \{0,1\}^{\alpha(k)}$ and a decomposable part $\mathsf{ots}^{\mathsf{dec}} \in \{0,1\}^k$.*
  - $\mathsf{dec\text{-}OTD}$: *On input a batch subset $K \subseteq [k]$, a partial sender message $\mathsf{ots}' \in \{0,1\}^{\alpha(k)+|K|}$, and an internal state st, $\mathsf{dec\text{-}OTD}$ outputs a vector of messages $(\widetilde{m}_i)_{i\in K} \in \{0,1\}^{|K|}$.*

– **Decomposable Correctness.** *For every* $\lambda \in \mathbb{N}^\star$*,* $K \subseteq [k]$*, every* $\vec{b} = (b_1, \ldots, b_k) \in \{0,1\}^k$*, and every* $\vec{m} = ((m_0^{(i)}, m_1^{(i)}))_{i\in[k]} \in \{0,1\}^{2k}$*,*

$$\Pr\left[(\widetilde{m}_1, \ldots, \widetilde{m}_{|K|}) = (m_{b_i}^{(i)})_{i\in K} : \begin{array}{c}(\mathsf{otr}, \mathsf{st}) \xleftarrow{\$} \mathsf{dec\text{-}OTR}(1^\lambda, \vec{b}) \\ (\mathsf{ots}^\star, \mathsf{ots}^{DB}) \xleftarrow{\$} \mathsf{dec\text{-}OTS}(1^\lambda, \mathsf{otr}, \vec{m}) \\ (\widetilde{m}_1, \ldots, \widetilde{m}_{|K|}) \xleftarrow{\$} \mathsf{dec\text{-}OTD}(K, (\mathsf{ots}^\star, [\mathsf{ots}^{\mathsf{dec}}]_K), \mathsf{st})\end{array}\right] = 1 \ .$$

– **Receiver Security (against Semi-Honest Sender).** *There exists an expected polynomial time simulator* Sim_S *such that for every* $\lambda \in \mathbb{N}^\star$ *and every* $\vec{b} = (b_1, \ldots, b_k) \in \{0,1\}^k$*,*

$$\left\{\mathsf{otr} : (\mathsf{otr}, \mathsf{st}) \xleftarrow{\$} \mathsf{dec\text{-}OTR}(1^\lambda, \vec{b})\right\} \overset{\mathrm{c}}{\approx} \left\{\mathsf{Sim}_S(1^\lambda)\right\} \ .$$

– **Decomposable Sender Security (against Semi-Honest Receiver).** *There exists an expected polynomial time simulator* Sim_R *such that for every* $\lambda \in \mathbb{N}^\star$*, every* $K \subseteq [k]$*, every* $\vec{b} = (b_1, \ldots, b_k) \in \{0,1\}^k$*, and every* $\vec{m} = ((m_0^{(i)}, m_1^{(i)}))_{i\in[k]} \in \{0,1\}^{2k}$*,*

$$\left\{(\mathsf{ots}^\star, [\mathsf{ots}^{\mathsf{dec}}]_K, \mathsf{otr}, \mathsf{st}) : \begin{array}{c}(\mathsf{otr}, \mathsf{st}) \xleftarrow{\$} \mathsf{dec\text{-}OTR}(1^\lambda, \vec{b}) \\ (\mathsf{ots}^\star, \mathsf{ots}^{BD}) \xleftarrow{\$} \mathsf{dec\text{-}OTS}(1^\lambda, \mathsf{otr}, \vec{m})\end{array}\right\} \overset{\mathrm{c}}{\approx}$$
$$\left\{(\mathsf{sim}^\star, \mathsf{sim}^{\mathsf{dec}}, \mathsf{otr}, \mathsf{st}) : \begin{array}{c}(\mathsf{otr}, \mathsf{st}) \xleftarrow{\$} \mathsf{dec\text{-}OTR}(1^\lambda, \vec{b}) \\ (\mathsf{sim}^\star, \mathsf{sim}^{\mathsf{dec}}) \xleftarrow{\$} \mathsf{Sim}_R(1^\lambda, K, (m_{b_i}^{(i)})_{i\in K}, \vec{b}, \mathsf{otr}, \mathsf{st})\end{array}\right\} \ .$$

3.2 Instantiation under QR + LPN, Adapted from [BBDP22]

As noted previously, two-round decomposable batch oblivious transfer can be seen as a strengthening of two-round batch OT with constant upload-rate and download-rate asymptotically one. As a matter of fact, the construction of batch OT with optimal rate from [BBDP22] natively satisfies the extra requirements and can be cast as two-round decomposable batch OT with sublinear overhead.

Theorem 5 (Corollary of [BBDP22, Section 7]). *Assume the* QR *assumption and the binary LPN assumption* $\mathsf{LPN}(\mathsf{dim}, \mathsf{num}, \rho)$ *with dimension* $\mathsf{dim} = \mathsf{poly}(\lambda)$*, number of samples* $\mathsf{num} = \mathsf{dim}^c$ *(for any constant* $c > 1$*), and noise rate* $\rho = \mathsf{num}^{\varepsilon-1}$ *(for some constant* $\varepsilon < 1$*). Then for any* $\ell = \ell(\lambda)$*, there exists a decomposable two-round batch oblivious transfer for batch size* $k = \ell \cdot \mathsf{num}$ *where*

– *The receiver message* otr *has size* $(\ell^2 \cdot \mathsf{dim} + \ell \cdot \mathsf{num}^\varepsilon) \cdot \mathsf{poly}(\lambda) + k$
– *The sender message* $\mathsf{ots} = (\mathsf{ots}^\star, \mathsf{ots}^{\mathsf{dec}})$ *has size* $|\mathsf{ots}^\star| = (\mathsf{num} + \ell \cdot \mathsf{num}^\varepsilon) \cdot \mathsf{poly}(\lambda)$ *and* $|\mathsf{ots}^{\mathsf{dec}}| = k$*.*

In particular, for appropriate parameters (sufficiently large ℓ*, and* num *sufficiently larger than* ℓ*),* $|\mathsf{otr}| = k + o(k)$*, and* $|\mathsf{ots}^\star| = o(k)$*.*

The proof of theorem 5 is deferred to the full version of this paper. Note that the construction of batch OT in [BBDP22] from LPN plus DDH or LPN plus polynomial-modulus LWE can similarly be shown to be decomposable. However, two-party sublinear secure computation is already known under these assumptions, via HSS for NC^1 [BGI16, BKS19].

3.3 Bounded Query Repetitions

At a high level the goal of this section is to show how a receiver message of $\mathsf{dec\text{-}OT}$ can be re-used, possibly with imbalances in how many times each selection bit in the batch is re-used, while asymptotically preserving upload- and download-rate.

Definition 6 (Decomposable Two-Round Batch Oblivious Transfer with Bounded Query Repetitions). *Let $k \in \mathbb{N}^\star$ and $\alpha = o(n)$. A semi-honest two-round decomposable batch OT protocol with $\alpha(\cdot)$-overhead and T-bounded query repetitions between a sender and a receiver can be defined as a triple of PPT algorithms $\mathsf{rep\text{-}OT} = (\mathsf{rep\text{-}OTR}, \mathsf{rep\text{-}OTS}, \mathsf{rep\text{-}OTD})$ with the following syntax and properties:*

- **Syntax.**
  - $\mathsf{rep\text{-}OTR}$: *On input the security parameter 1^λ and a vector of selection bits $\vec{b} = (b_1, \ldots, b_k) \in \{0,1\}^k$, $\mathsf{rep\text{-}OTR}$ outputs a receiver message $\mathsf{otr} \in \{0,1\}^{\mathcal{O}(k)}$ and an internal state st; without loss of generality we assume that st contains all the random coins used by $\mathsf{rep\text{-}OTR}$ as well as $\vec{b}$.*
  - $\mathsf{rep\text{-}OTS}$: *On input the security parameter 1^λ, a query otr, a database $((m_0^{(i)}, m_1^{(i)}))_{i\in[k']} \in \{0,1\}^{2k'}$ (where $k \le k' \le k \cdot T$), and a vector of repetitions $\mathsf{rep} = (\mathsf{rep}_1, \ldots, \mathsf{rep}_k) \in [0,T]^k$ such that $\sum_{i=1}^k \mathsf{rep}_i = k'$, $\mathsf{rep\text{-}OTS}$ outputs a sender message $\mathsf{ots} = (\mathsf{ots}^\star, \mathsf{ots}^{\mathsf{dec}})$, which is comprised of a reusable part $\mathsf{ots}^\star \in \{0,1\}^{\alpha(k)}$ and a decomposable part $\mathsf{ots}^{\mathsf{dec}} \in \{0,1\}^{k'}$, as well as rep.*
  - $\mathsf{rep\text{-}OTD}$: *On input a batch subset $K \subseteq [k']$, a partial sender message $\mathsf{ots}' \in \{0,1\}^{\alpha(k)+|K|}$, a vector of repetitions $\mathsf{rep} = (\mathsf{rep}_1, \ldots, \mathsf{rep}_k) \in [0,T]^k$ such that $\sum_{i=1}^k \mathsf{rep}_i = k'$, and an internal state st, $\mathsf{rep\text{-}OTD}$ outputs a vector of messages $(\widetilde{m}_i)_{i\in K} \in \{0,1\}^{|K|}$.*
- **Decomposable Correctness.** *For every $\lambda \in \mathbb{N}^\star$, $K \subseteq [k']$, every $\vec{b} = (b_1, \ldots, b_k) \in \{0,1\}^k$, and every $\vec{m} = ((m_0^{(i)}, m_1^{(i)}))_{i\in[k']} \in \{0,1\}^{2k'}$,*

$$\Pr\left[(\widetilde{m}_1, \ldots, \widetilde{m}_{|K|}) = (m_{\sigma_i}^{(i)})_{i\in K} \colon \begin{array}{r}(\mathsf{otr}, \mathsf{st}) \xleftarrow{\$} \mathsf{rep\text{-}OTR}(1^\lambda, \vec{b}) \\ ((\mathsf{ots}^\star, \mathsf{ots}^{\mathsf{dec}}), \mathsf{rep}) \xleftarrow{\$} \mathsf{rep\text{-}OTS}(1^\lambda, \mathsf{otr}, \vec{m}, \mathsf{rep}) \\ (\widetilde{m}_1, \ldots, \widetilde{m}_{|K|}) \xleftarrow{\$} \mathsf{rep\text{-}OTD}(K, (\mathsf{ots}^\star, [\mathsf{ots}^{\mathsf{dec}}]_K), \mathsf{rep}, \mathsf{st})\end{array}\right] = 1 \ ,$$

$$\text{where } \sigma_i \coloneqq b_{\max\{j \colon (\sum_{j'<j} \mathsf{rep}_{j'}) \le i\}} \ .$$

- **Receiver Security (against Semi-Honest Sender).** *There exists an expected polynomial time simulator Sim_S such that for every $\lambda \in \mathbb{N}^\star$ and every $\vec{b} = (b_1, \ldots, b_k) \in \{0,1\}^k$,*

$$\left\{\mathsf{otr} \colon (\mathsf{otr}, \mathsf{st}) \xleftarrow{\$} \mathsf{rep\text{-}OTR}(1^\lambda, \vec{b})\right\} \overset{\mathrm{c}}{\approx} \left\{\mathsf{Sim}_S(1^\lambda)\right\} \ .$$

- **Decomposable Sender Security (against Semi-Honest Receiver).** *There exists an expected polynomial time simulator Sim_R such that for every $\lambda \in \mathbb{N}^\star$, every $\mathsf{rep} = (\mathsf{rep}_1, \ldots, \mathsf{rep}_k) \in [0,T]^k$ such that $\|\mathsf{rep}\|_1 = k'$, every*

$K \subseteq [k']$, *every* $\vec{b} = (b_1, \ldots, b_k) \in \{0,1\}^k$, *and every* $\vec{m} = ((m_0^{(i)}, m_1^{(i)}))_{i \in [k']} \in \{0,1\}^{2k'}$,

$$\left\{ (\mathsf{ots}^\star, [\mathsf{ots}^{\mathsf{dec}}]_K, \mathsf{otr}, \mathsf{st}) : \begin{array}{c} (\mathsf{otr}, \mathsf{st}) \xleftarrow{\$} \mathsf{rep\text{-}OTR}(1^\lambda, \vec{b}) \\ (\mathsf{ots}^\star, \mathsf{ots}^{\mathsf{dec}}) \xleftarrow{\$} \mathsf{rep\text{-}OTS}(1^\lambda, \mathsf{otr}, \vec{m}, \mathsf{rep}) \end{array} \right\} \overset{\mathrm{c}}{\approx}$$

$$\left\{ (\mathsf{sim}^\star, \mathsf{sim}^{\mathsf{dec}}, \mathsf{otr}, \mathsf{st}) : \begin{array}{c} (\mathsf{otr}, \mathsf{st}) \xleftarrow{\$} \mathsf{rep\text{-}OTR}(1^\lambda, \vec{b}) \\ (\mathsf{sim}^\star, \mathsf{sim}^{\mathsf{dec}}) \xleftarrow{\$} \mathsf{Sim}_R(1^\lambda, K, (m_{\sigma_i}^{(i)})_{i \in K}, \vec{b}, \mathsf{rep}, \mathsf{otr}, \mathsf{st}) \end{array} \right\}$$

where $\sigma_i := b_{\max\{j \colon (\sum_{j' < j} \mathsf{rep}_{j'}) \leq i\}}$.

Decomposable Two-Round Batch Oblivious Transfer with Bounded Query Repetition

Parameters: Batch number k, Repetition bound T.
Requires: A two-round decomposable batch dec-OT protocol $\mathsf{dec\text{-}OT} = (\mathsf{dec\text{-}OTR}, \mathsf{dec\text{-}OTS}, \mathsf{dec\text{-}OTD})$ with α-overhead such that $\alpha(k) = o(k/T)$.

rep-OTR: On input the security parameter 1^λ and a vector of selection bits $\vec{b} = (b_1, \ldots, b_k) \in \{0,1\}^k$:

1. Compute $(\mathsf{otr}, \mathsf{st}) \xleftarrow{\$} \mathsf{dec\text{-}OTR}(1^\lambda, \vec{b})$.
2. Output $(\mathsf{otr}, \mathsf{st})$.

rep-OTS: On input the security parameter 1^λ, a receiver message otr, a database $\vec{m} \in \{0,1\}^{2k'}$, and a vector of repetitions $\mathsf{rep} = (\mathsf{rep}_1, \ldots, \mathsf{rep}_k) \in [0,T]^k$ such that $\|\mathsf{rep}\|_1 = k'$:

1. Parse $\vec{m}$ as $((m_0^{(j,i)}, m_1^{(j,i)}))_{j \in [k], i \in [\mathsf{rep}_j]}$.
2. For $j \in [k]$ and $i \in [\mathsf{rep}_j + 1, T]$, set $(m_0^{(j,i)}, m_1^{(j,i)}) \leftarrow (0,0)$.
3. For $i \in [T]$ set $\vec{m}_i := ((m_0^{(j,i)}, m_1^{(j,i)}))_{j \in [k]}$.
4. For $j = 1 \ldots T$:
   Compute $(\mathsf{ot}_{S,j}^\star, \mathsf{ot}_{S,j}^{\mathsf{dec}}) \xleftarrow{\$} \mathsf{dec\text{-}OTS}(1^\lambda, \mathsf{otr}, \vec{m}_j)$.
5. Set $\mathsf{ots}^\star \leftarrow \mathsf{ot}_{S,1}^\star \| \ldots \| \mathsf{ot}_{S,T}^\star$ and $\mathsf{ot}_S^{\mathsf{dec}} \leftarrow ([\mathsf{ot}_{S,1}^{\mathsf{dec}}]_1 \| \ldots \| [\mathsf{ot}_{S,\mathsf{rep}_1}^{\mathsf{dec}}]_1) \| \ldots \| ([\mathsf{ot}_{S,1}^{\mathsf{dec}}]_k \| \ldots \| [\mathsf{ot}_{S,\mathsf{rep}_k}^{\mathsf{dec}}]_k)$.
6. Output $(\mathsf{ot}_S^\star, \mathsf{ot}_S^{\mathsf{dec}})$.

rep-OTD: On input a sender message ots, a vector of repetitions $\mathsf{rep} = (\mathsf{rep}_1, \ldots, \mathsf{rep}_k) \in [0,T]^k$ such that $\|\mathsf{rep}\|_1 = k'$, and an internal state st:

1. Parse ots as $(\mathsf{ot}_S^\star, \mathsf{ot}_S^{\mathsf{dec}})$.
2. Parse $\mathsf{ot}_S^\star$ as $\mathsf{ot}_{S,1}^\star \| \ldots \| \mathsf{ot}_{S,T}^\star$.
3. Parse $\mathsf{ot}_S^{\mathsf{dec}}$ as $([\mathsf{ot}_{S,1}^{\mathsf{dec}}]_1 \| \ldots \| [\mathsf{ot}_{S,\mathsf{rep}_1}^{\mathsf{dec}}]_1) \| \ldots \| ([\mathsf{ot}_{S,1}^{\mathsf{dec}}]_k \| \ldots \| [\mathsf{ot}_{S,\mathsf{rep}_k}^{\mathsf{dec}}]_k)$.

4. For $i = 1 \dots T$:
   (a) Set $K_i \coloneqq \{j \colon j \in [k], \mathsf{rep}_j \leq i\}$, ordered according to the natural order on $\mathbb{N}$.
   (b) Set $\vec{v}_i \leftarrow \|_{j \in K_i} [\mathsf{ot}^{\mathsf{dec}}_{S,i}]_j$.
   (c) Compute $(\widetilde{m}_{i,j})_{j \in K_i} \xleftarrow{\$} \mathsf{dec\text{-}OTD}(K_i, (\mathsf{ot}^{\star}_{S,i}, v_i), \mathsf{st})$.
5. Output $(\widetilde{m}_{1,1}, \dots, \widetilde{m}_{\mathsf{rep}_1,1}, \widetilde{m}_{1,2}, \dots, \widetilde{m}_{\mathsf{rep}_2,2}, \dots, \widetilde{m}_{\mathsf{rep}_k,k})$.

Fig. 1: From dec-OT with α overhead to rep-OT with $\alpha \cdot T$ overhead.

Lemma 7 (From dec-OT to rep-OT). *If* dec-OT *is a semi-honest two-round decomposable batch OT protocol with α overhead, then the construction* rep-OT *from fig. 1 is a semi-honest two-round decomposable batch OT protocol with $\alpha \cdot T$ overhead and T-bounded repetitions.*

The proof of lemma 7 is deferred to the full version of this paper.

3.4 Two-Round Batch SPIR with Correlated Queries

We next introduce and achieve a notion of batch symmetric PIR with correlated queries. This corresponds to batch SPIR where the queries are not independent; rather, the total entropy w used to describe the queries is small, and the queried indices can be reconstructed via a public function that "mixes and matches" the individual bits of entropy $\vec{\alpha} = (\alpha_1, \dots, \alpha_w)$ in a public manner. This will allow us to compress k size-N databases each down to a single bit—achieving batch SPIR—using batch OT on the selection bit vector $\vec{\alpha}$, by building a "Merkle-like forest" (seeing each 1-out-of-2 OT in the batch as a roughly length-halving hash function): the correlation in the queries across different databases is what allows the nodes of the Merkle forest to be batched.

In more detail, if the w bits of entropy are $\alpha_1, \dots, \alpha_w$, "mixing and matching" means that each of the ($n = \log N$)-bit queries to a single database can be obtained by concatenating n of the bits α_i, possibly permuted. In the notation below, the j^{th} query is given by vector $(\alpha_{s_{j,1}}, \dots, \alpha_{s_{j,n}})$. We will be interested in how many times a given α_i appears within the k queries (counted by the occurrence function t_i below), as well as how many times it appears in specific position $j' \in [n]$ within the k queries (denoted below by $t_{i,j'}$). If all $t_{i,j'}$ are bounded by T, then for each level $j' \in [n]$ in the "Merkle forest" we can achieve the desired length-halving compression by using at most T batch OT sender computations on the original batch OT selection vector $\vec{\alpha}$.

Definition 8 ("Mix and Match" Functions). *A "mix and match" function* $\mathsf{MixAndMatch} \colon \{0,1\}^w \to [N]^k$ *is one parameterised by k ordered subsets of $n \coloneqq \log N$ elements of $[w]$, $S_j = (s_{j,1}, \dots, s_{j,n}) \in [w]^n$ for $j \in [k]$ such that:*

$$\forall \vec{\alpha} = (\alpha_1, \dots, \alpha_w) \in \{0,1\}^w, \mathsf{MixAndMatch}(\alpha_1, \dots, \alpha_w) \coloneqq (x_1, \dots, x_k), \\ \text{with } x_j \coloneqq \alpha_{s_{j,1}} \cdots \alpha_{s_{j,n}} \in [N].$$

Such a function is associated with an occurrence function, which counts the occurrences of each input position in the outputs:

$$\begin{aligned} t_{\cdot}\colon\ [w] &\to [k] \\ i\ &\mapsto t_i = \sum_{j=1}^{k} \mathbf{1}_{i \in S_j} \end{aligned}$$

Each t_i ($i \in [w]$) can be decomposed as $t_i = t_{i,1} + \cdots + t_{i,n}$, where $t_{i,j'}$ is equal to the number of values of $j \in [k]$ such that $s_{j,j'} = i$.

- *MixAndMatch is said to be* T-balanced *if* $\forall i \in [w], \forall j' \in [n], t_{i,j'} \leq T$.
- *MixAndMatch is said to be* T-balanceable *if it can be expressed as the function* $\mathsf{MixAndMatch} = (\mathsf{MixAndMatch}' \circ \mathsf{replicate})$, *where* $\mathsf{MixAndMatch}\colon \{0,1\}^{w'} \to [N]^k$ *is a* T*-balanced mix-and-match function and* replicate *is defined as:*

$$\begin{aligned} \mathsf{replicate}\colon\quad \{0,1\}^w &\to \{0,1\}^{w'} \\ (b_1, \ldots, b_w) &\mapsto (b_1^{\|\lceil t_1/T \rceil} \| \ldots \| b_w^{\|\lceil t_w/T \rceil}) \end{aligned} \qquad \text{where } w' := \sum_{i \in [w]} \lceil t_i/T \rceil.$$

Lemma 9. *Let $w, n \in \mathbb{N}$ be a sufficiently large integers. For any family of unordered subsets $S_1, \ldots, S_k \in \binom{[w]}{n}$ there exists an ordering of each subset S_j such that the mix-and-match function induced by the resulting $(\tilde{S}_j)_{j \in [k]}$ is* $\mathsf{polylog}(w)$*-balanceable.*
Furthermore, such orderings can be found in expected constant time.

The proof of lemma 9 is deferred to the full version of this paper.

Definition 10 (Two-Round Batch Computational Batch SPIR with Correlated "Mix and Match" Queries). *A semi-honest two-round batch SPIR protocol with correlated "mix and match" queries between a sender and a receiver can be defined as a triple of PPT algorithms* $\mathsf{corrSPIR} = (\mathsf{corrSPIR}_R, \mathsf{corrSPIR}_S, \mathsf{corrSPIR}_D)$ *parameterised by a public* T-balanceable *"mix and match" function (definition 8)* $\mathsf{MixAndMatch}\colon \{0,1\}^w \to [N]^k$ *with the following syntax and properties:*

- **Syntax.**
  - $\mathsf{corrSPIR}_R$: *On input the security parameter* 1^λ *and a vector of selection bits* $\vec{b} = (b_1, \ldots, b_w) \in \{0,1\}^w$, $\mathsf{corrSPIR}_R$ *outputs a receiver message* $\mathsf{spir}_R \in \{0,1\}^{\mathcal{O}(w)}$ *and an internal state* st*; without loss of generality, we assume* st *contains all the coins used by* $\mathsf{corrSPIR}_R$ *as well as* $\vec{b}$.
  - $\mathsf{corrSPIR}_S$: *On input the security parameter* 1^λ, *a receiver message* spir_R, *and* k N*-bit databases* $\vec{m}_1, \ldots, \vec{m}_k \in \{0,1\}^N$, $\mathsf{corrSPIR}_S$ *outputs a sender message* $\mathsf{spir}_S \in \{0,1\}^{\mathcal{O}(k)}$.
  - $\mathsf{corrSPIR}_D$: *On input a sender message* spir_S *and an internal state* st, $\mathsf{corrSPIR}_D$ *outputs a vector of messages* $(\widetilde{m}_1, \ldots, \widetilde{m}_k) \in \{0,1\}^k$.

– **Correctness.**

$$\forall \vec{b} = (b_1, \ldots, b_w) \in \{0,1\}^w, \forall \vec{M} = (\vec{m}_1, \ldots, \vec{m}_k) \in \{0,1\}^{N \cdot k},$$
$$\Pr\left[(\widetilde{m}_1, \ldots, \widetilde{m}_k) = (\vec{m}_1[x_1], \ldots, \vec{m}_k[x_k]) : \begin{array}{r}(\mathsf{spir}_R, \mathsf{st}) \xleftarrow{\$} \mathsf{corrSPIR}_R(1^\lambda, \vec{b}) \\ \mathsf{spir}_S \xleftarrow{\$} \mathsf{corrSPIR}_S(1^\lambda, \mathsf{spir}_R, \vec{M}) \\ (\widetilde{m}_1, \ldots, \widetilde{m}_k) \xleftarrow{\$} \mathsf{corrSPIR}_D(\mathsf{spir}_S, \mathsf{st})\end{array}\right] = 1$$
$$\text{where } (x_1, \ldots, x_k) \coloneqq \mathsf{MixAndMatch}(\vec{b}).$$

– **Security.** *The following protocol securely realises* $\mathcal{F}_{\mathsf{corrSPIR}}$ *(fig. 2) in the presence of a semi-honest adversary: the receiver computes* $(\mathsf{spir}_R, \mathsf{st}) \xleftarrow{\$} \mathsf{corrSPIR}_R(1^\lambda, \vec{b})$ *and sends* spir_R *to the sender, who in turn computes* $\mathsf{spir}_S \xleftarrow{\$} \mathsf{corrSPIR}_S(1^\lambda, \mathsf{spir}_R, \vec{M})$ *and returns* spir_S*; finally, the receiver computes and outputs* $(\widetilde{m}_1, \ldots, \widetilde{m}_k) \xleftarrow{\$} \mathsf{corrSPIR}_D(\mathsf{spir}_S, \mathsf{st})$.

Functionality $\mathcal{F}_{\mathsf{corrSPIR}}$

The functionality $\mathcal{F}_{\mathsf{corrSPIR}}$ is parameterised by the number k of SPIRs in the batch, the size N of each database, and the number w of selection bits. Furthermore, it is parameterised by a public T-balanceable "mix and match" function (definition 8) $\mathsf{MixAndMatch}\colon \{0,1\}^w \to [N]^k$. $\mathcal{F}_{\mathsf{corrSPIR}}$ interacts with an ideal sender $\mathbf{S}$ and an ideal receiver $\mathbf{R}$ via the following queries.

1. On input $(\texttt{sender}, \vec{M} = (\vec{m}_i)_{i \in [k]})$ from $\mathbf{S}$, with $\vec{m}_i = (m_{i,j})_{j \in [N]} \in \{0,1\}^N$ store $\vec{M}$.
2. On input $(\texttt{receiver}, (b_j)_{j \in [w]})$ from $\mathbf{R}$, check if a tuple of inputs $\vec{M}$ has already been recorded; if so, compute $(x_1, \ldots, x_k) \coloneqq \mathsf{MixAndMatch}(b_1, \ldots, b_w) \in [N]^k$, send $(m_{i,x_i})_{i \in [k]}$ to $\mathbf{R}$, and halt.

If the functionality receives an incorrectly formatted input, it aborts.

Fig. 2: Ideal Functionality $\mathcal{F}_{\mathsf{corrSPIR}}$ for Batch SPIR with Correlated "Mix and Match" Queries

Batch SPIR with Correlated "Mix and Match" Queries

Parameters: k, N, $n \coloneqq \log N$, w, T, a T-balanceable $\mathsf{MixAndMatch}\colon \{0,1\}^w \to [N]^k$ (parameterised by subsets $S_j = (s_{j,1}, \ldots, s_{j,n}) \in [w]^n$ for $j \in [k]$) and an associated list of number of occurrences $(t_1, \ldots, t_w)$ with $t_i = t_{i,1} + \cdots + t_{i,n}$, a two-round batch rep-OT protocol $\mathsf{rep\text{-}OT} = (\mathsf{rep\text{-}OTR}, \mathsf{rep\text{-}OTS}, \mathsf{rep\text{-}OTD})$.

$\mathsf{corrSPIR}_R$**:** On input the security parameter 1^λ and a vector of selection bits $\vec{b} = (b_1, \ldots, b_w) \in \{0,1\}^w$:

1. Set $\vec{b}' \leftarrow b_1^{\|\lceil t_1/T\rceil} \| \dots \| b_w^{\|\lceil t_w/T\rceil}$.
2. Compute $(\mathsf{spir}_R, \mathsf{st}) \xleftarrow{\$} \mathsf{OTR}(1^\lambda, \vec{b}')$, and output $(\mathsf{spir}_R, \mathsf{st} \| \vec{b})$.

$\mathsf{corrSPIR}_S$**:** On input the security parameter 1^λ, a receiver message spir_R, and k databases $\vec{m}_1, \dots, \vec{m}_k \in [N]$:

1. Set $(\mathtt{DB}_{1,1}, \dots, \mathtt{DB}_{1,k}) \coloneqq (\vec{m}_1, \dots, \vec{m}_k)$.
2. For $d = 1, \dots, n$:
   (a) For $i = 1, \dots, w$:
   $$\text{Set } \mathsf{rep}_{d,i} \leftarrow \begin{cases} T^{\|\lceil t_{i,d}/T\rceil} \| 0^{\|\lceil t_i/T\rceil - \lceil t_{i,d}/T\rceil} & \text{if } T | t_{i,d} \\ T^{\|\lfloor t_{i,d}/T\rfloor} \| t_{i,d} \% T \| 0^{\|\lceil t_i/T\rceil - \lceil t_{i,d}/T\rceil} & \text{if } T \nmid t_{i,d} \end{cases}$$
   (b) Set $\mathsf{rep}_d \leftarrow \mathsf{rep}_{d,1} \| \dots \| \mathsf{rep}_{d,w}$.
   (c) Initialise $X_d \leftarrow \varnothing$.
   (d) For $j = 1, \dots, k$:
   For $x = 0, \dots, N/2^d - 1$:
   $X_d.\mathsf{append}(((\mathtt{DB}_{d,j}[2x], \mathtt{DB}_{d,j}[2x+1]),\ s_{j,d},\ x,\ j))$.
   (e) Sort X_d according to the lexicographic order which first sorts by increasing fourth element (the "$j \in [k]$") and then, in case of equality, by increasing third element (the "$x \in [0, N/2^d - 1]$").
   (f) Greedily partition X_d as $X_d = X_{d,1} \sqcup \dots \sqcup X_{d,(N/2^d)}$ such that for each $\ell \in [N/2^d]$ and each $i \in [w]$, $X_{d,\ell}$ contains (up to) $t_{i,d}$ elements of the form $((\cdot,\cdot), i, \cdot, \cdot)$; "greedily" is here taken to mean that the first $t_{i,d}$ elements of the form $((\cdot,\cdot), i, \cdot, \cdot)$ are placed in $X_{d,1}$, the next $t_{i,d}$ in $X_{d,2}$, and so on.
   (g) For $\ell = 1, \dots, N/2^d$:
   – Sort $X_{d,\ell}$ according to the second element in increasing order, breaking ties with the fourth, and then if necessary the third element of the 4-tuples.
   – Set $\mathtt{DB}'_{d,\ell} \leftarrow (S_{d,\ell}[0].\mathsf{first}, \dots, S_{d,\ell}[(\sum_{i=1}^{w} t_{i,d}) - 1].\mathsf{first}) \in \{0,1\}^{2|S_{d,\ell}|}$.
   – Set $(\mathsf{ots}^\star_{d,\ell}, \mathsf{ots}^{\mathsf{dec}}_{d,\ell}) \xleftarrow{\$} \mathsf{rep\text{-}OTS}(1^\lambda, \mathsf{spir}_R, \mathtt{DB}'_{d,\ell}, \mathsf{rep}_d)$.
   (h) If $d < n$:
   – For $j = 1, \dots, k$:
   Initialise $\mathtt{DB}_{d+1,j} \leftarrow 0^{\|N/2^d}$.
   – For $\ell = 1, \dots, N/2^d$:
   For $\ell' = 0, \dots, (\sum_{i=1}^{w} t_{i,d}) - 1$:
   Parse $X_{d,\ell}[\ell']$ as $((\cdot,\cdot), \cdot, x, j)$, with $x \in [N/2^d]$ and $j \in [k]$.
   Set $\mathtt{DB}_{d+1,j}[x] \leftarrow \mathsf{ots}^{\mathsf{dec}}_{d,\ell}[\ell']$.
   (i) Set $\mathsf{ots}^\star_d \leftarrow (\mathsf{ots}^\star_{d,1}, \dots, \mathsf{ots}^\star_{d,N/2^d})$.
3. Set $\mathsf{spir}_S \coloneqq ((\mathsf{ots}^\star_1, \dots, \mathsf{ots}^\star_n), \mathsf{ots}^{\mathsf{dec}}_n)$, and output spir_S.

$\mathsf{corrSPIR}_D$**:** On input a sender message spir_S and an internal state st:

1. Parse spir_S as $\mathsf{spir}_S = ((\mathsf{ots}^\star_1, \ldots, \mathsf{ots}^\star_n), \mathsf{ots}^{\mathsf{dec}}_n)$, and parse st as $\mathsf{st}' \| \vec{b}$.
2. Set $(y_1, \ldots, y_k) \leftarrow \mathsf{MixAndMatch}(\vec{b})$ (*i.e.* $y_j \leftarrow b_{s_{j,1}} \ldots b_{s_{j,n}}$ for $j \in [n]$).
3. Initialise $(\widetilde{m}_1, \ldots, \widetilde{m}_k) \leftarrow \mathsf{ots}^{\mathsf{dec}}_n$.
4. For $d = 1, \ldots, n$:
   (a) Initialise $X_d \leftarrow \varnothing$.
   (b) Initialise $X_d \leftarrow ((\perp,\ s_{j,d},\ x,\ j))_{j \in [k], x \in [0, N/2^d - 1]}$
   (c) Sort X_d according to the lexicographic order which first sorts by increasing fourth element (the "$j \in [k]$") and then, in case of equality, by increasing third element (the "$x \in [0, N/2^d - 1]$").
   (d) Greedily partition X_d as $X_d = X_{d,1} \sqcup \cdots \sqcup X_{d,N/2^d}$ such that for each $\ell \in [N/2^d]$ and each $i \in [w]$, $X_{d,\ell}$ contains exactly $t_{i,d}$ elements of the form $(\cdot, i, \cdot, \cdot)$; "greedily" is here taken to mean that the first $t_{i,d}$ elements of the form $(\cdot, i, \cdot, \cdot)$ are placed in $X_{d,1}$, the next $t_{i,d}$ in $X_{d,2}$, and so on.
   (e) For $\ell = 1, \ldots, N/2^d$:
       Sort $X_{d,\ell}$ according to the second element in increasing order, breaking ties with the fourth, and then if necessary the third element of the 4-tuples.
   (f) Parse $\mathsf{ots}^\star_d$ as $\mathsf{ots}^\star_d = (\mathsf{ots}^\star_{d,1}, \ldots, \mathsf{ots}^\star_{d,N/2^d})$
   (g) For $j = 1, \ldots, k$:
       – Set $\ell_{j,d}$ to be the unique $\ell \in [N/2^d]$ such that $(\perp,\ s_{j,d},\ (b_{s_{j,n}} \ldots b_{s_{j,d}}),\ j) \in X_{d,\ell}$.
       – Set $\mathsf{ind}_{j,d}$ to be the index of $(\perp,\ s_{j,d},\ (b_{s_{j,n}} \ldots b_{s_{j,d}}),\ j)$ in $X_{d,\ell}$.
       – Update $\widetilde{m}_j \leftarrow \mathsf{rep\text{-}OTD}(\{\mathsf{ind}_{j,d}\}, (\mathsf{ots}^\star_{d,\ell_{j,d}}, \widetilde{m}_j), \mathsf{rep}, \mathsf{st})$
5. Output $(\widetilde{m}_1, \ldots, \widetilde{m}_k)$.

Fig. 3: corrSPIR from rep-OT.

Theorem 11. *Assume that* rep-OT *is a semi-honest two-round decomposable batch OT protocol with $\alpha(\cdot)$-overhead and T-bounded query repetitions. Then construction* $(\mathsf{corrSPIR}_R, \mathsf{corrSPIR}_S, \mathsf{corrSPIR}_D)$ *from fig. 3 is a two-round batch SPIR protocol with correlated "mix and match" queries. Furthermore the size of the receiver message is linear in $w + k \cdot n/T$ and the size of the sender message is upper bounded by $k + (\log N) \cdot (N-1) \cdot \alpha(w + k \cdot n/T)$ (where k is the batch number and N is the size of each of the k databases).*

The proof of theorem 11 is deferred to the full version of this paper.

3.5 Sublinear Computation of log log-Depth Circuits from corrSPIR

In this section theorem 12 shows how to build sublinear secure computation for shallow (roughly log log-depth) circuits from corrSPIR, with an explicit protocol provided in fig. 4. Main theorem 1 combines all of the previous theorems and

shows that sublinear secure computation for shallow circuits can be based on $\mathsf{QR} + \mathsf{LPN}$.

Protocol Π_{2PC}

Functionality:

- **Parameters:** $C\colon \{0,1\}^n \to \{0,1\}^m$ is a boolean circuit of depth k. For $j \in [m]$, $S_j = \{s_{j,1}, \ldots, s_{j,2^k}\}$ is the subset[a] of the inputs on which depends the j^{th} output of f, and for $i \in [n]$ we denote t_i the number of outputs of C on which the i^{th} variable depends. $(\pi_j)_{j\in[m]} \in (\mathfrak{S}_{2^k})^m$ is a family of m permutations on $[2^k]$, such that the following is a $(T = \mathsf{polylog}(n))$-balanced "mix and match" function:

$$\begin{array}{rccc} \mathsf{MixAndMatch}_C\colon & \{0,1\}^w & \to & [2^k]^m \\ & (x_1, \ldots, x_w) & \mapsto & (x_{s_{j,\pi_j(1)}} \| \ldots \| x_{s_{j,\pi_j(2^k)}})_{j\in[m]} \end{array}$$

- **Inputs:** Parties P_0 and P_1 hold additive shares $(\vec{x}_0, \vec{x}_1)$ of an input $\vec{x} \in \{0,1\}^n$.
- **Outputs:** The parties output $C(\vec{x})$.
- **Requires:** $\mathsf{corrSPIR} = (\mathsf{corrSPIR}_R, \mathsf{corrSPIR}_S, \mathsf{corrSPIR}_D)$ is a two-round batch SPIR protocol with correlated "mix and match" queries.

Protocol:

1. P_0 samples $\vec{y}_0 \xleftarrow{\$} \{0,1\}^m$ and for $j \in [m]$ sets $\mathtt{DB}_j \in \{0,1\}^{2^{2^k}}$ to be the truth table of the following function:

$$\begin{array}{rccc} g_j\colon & \{0,1\}^{2^k} & \to & \{0,1\} \\ & (X_1, \ldots, X_{2^k}) & \mapsto & C_j((X_{\pi_j(1)} \oplus \vec{x}_0[\pi_j(1)] \| \ldots \| X_{\pi_j(2^k)} \oplus \vec{x}_0[\pi_j(2^k)])) \oplus \vec{y}_0[j] \end{array}$$

   where C_j is the j^{th} output of C.

2. P_1 sets $\vec{x}'_1 \leftarrow (\vec{x}_1[1])^{\| \lceil t'_1/T \rceil} \| \ldots \| b_w^{\| \lceil t'_w/T \rceil}$.
3. P_1 samples $(\mathsf{spir}_R, \mathsf{st}) \xleftarrow{\$} \mathsf{corrSPIR}_R(1^\lambda, \vec{x}_1)$ and sends spir_R to P_0.
4. P_0 samples $\mathsf{spir}_S \xleftarrow{\$} \mathsf{corrSPIR}_S(1^\lambda, \mathsf{spir}_R, (\mathtt{DB}_j)_{j\in[m]})$ and sends $(\mathsf{spir}_S, \vec{y}_0)$ to P_1.
5. P_1 recovers $\vec{y}_1 \leftarrow \mathsf{corrSPIR}_D(\mathsf{spir}_S, \mathsf{st})$.
6. P_1 sets $\vec{y} \leftarrow \vec{y}_0 \oplus \vec{y}_1$, and sends $\vec{y}$ to P_0.
7. Each party P_σ outputs $\vec{y}$.

[a] Because C has depth k and each of its gate has fan-in at most 2, each output value only depends on at most 2^k inputs. Without loss of generality we can assume each output depends on exactly 2^k (by allowing for trivial "dependencies").

Fig. 4: **Secure Computation of Low-Depth Circuits from** $\mathsf{corrSPIR}$

Theorem 12. *If* corrSPIR *is a two-round batch* SPIR *protocol with correlated "mix and match" queries, then* Π_{2PC} *from fig. 4 securely computes the randomized functionality* $(\vec{x}_0, \vec{x}_1) \mapsto \{(\vec{r}, C(\vec{x}_0 \oplus \vec{x}_1) \oplus \vec{r}) \colon \vec{r} \xleftarrow{\$} \{0,1\}^m\}$ *in the presence of a semi-honest adversary corrupting (at most) one of the two parties.*

The proof of theorem 12 is deferred to the full version of this paper.

Our first main theorem follows from the combination of theorem 5—which instantiates dec-OT from QR+LPN—, lemma 7—which provides a construction of rep-OT from dec-OT—, theorem 11—which provides a construction of corrSPIR from rep-OT, and theorem 12—which provides a secure computation protocol from corrSPIR.

Main Theorem 1 (Sublinear Secure Computation from $\mathsf{QR} + \mathsf{LPN}$)**.** *Assume the* QR *assumption and the binary LPN assumption* $\mathsf{LPN}(\mathsf{dim}, \mathsf{num}, \rho)$ *with dimension* $\mathsf{dim} = \mathsf{poly}(\lambda)$*, number of samples* $\mathsf{num} = (n+m)^{1/3} \cdot \mathsf{poly}(\lambda)$*, and noise rate* $\rho = \mathsf{num}^{\varepsilon-1}$ *(for some constant* $\varepsilon < 1$*). Then for any* n*-input* m*-output boolean circuit* C *of size* s *and depth* k*, there is a two-party protocol for securely computing* C *using only* $\mathcal{O}(n + m + 2^{k+2^k} \cdot \mathsf{polylog}(n) \cdot \mathsf{poly}(\lambda) \cdot ((n+m)^{2/3} + (n+m)^{(1+2\varepsilon)/3}))$ *bits of communication, and computation* $\mathsf{poly}(\lambda, 2^{2^k})$*.*

The discussion on the parameters is deferred to the full version of this paper.

Corollary 13 (Sublinear Secure Computation of $\log\log$**-Depth Circuits).** *Assume the* QR *assumption and the binary LPN assumption* $\mathsf{LPN}(\mathsf{dim}, \mathsf{num}, \rho)$ *with dimension* $\mathsf{dim} = \mathsf{poly}(\lambda)$*, number of samples* $\mathsf{num} = (n+m)^{1/3} \cdot \mathsf{poly}(\lambda)$*, and noise rate* $\rho = \mathsf{num}^{-1/2}$*. Then for any* n*-input* m*-output boolean circuit* C *of polynomial size* s *and depth* $\log\log s/4$*, there is a two-party protocol for securely computing* C *using only* $\mathcal{O}(n+m+\sqrt{s} \cdot \mathsf{poly}(\lambda) \cdot (n+m)^{2/3})$ *bits of communication, and polynomial computation.*

3.6 Extension to Layered Circuits

Layered circuits are boolean circuits whose gates can be arranged into layers such that any wire connects adjacent layers. It is well-known from previous works [BGI16, Cou19, CM21] that sublinear protocols for low-depth circuits translate to sublinear protocols for general layered circuits: the parties simply cut the layered circuit into low-depth "chunks", and securely evaluate it chunk-by-chunk. We refer to the full version of this paper for the extension of our protocol to layered circuits.

4 Polylogarithmic PIR from CDH

A private information retrieval is a two-party protocol between a server S holding a string z (the database) and a client C holding an integer i, where only the client

receives an output. The security parameter λ and the length $n(\lambda) = \mathsf{poly}(\lambda) = |z|$ of the server database are a common (public) input. We let $\mathsf{View}_S(\lambda, z, i)$ denote the view of S during its interaction with C on respective inputs (z, i) with common input $(\lambda, n = |z|)$, and by $\mathsf{Out}_C(\lambda, z, i)$ the output of C after the interaction.

Definition 14 (Private Information Retrieval). *A private information retrieval for database size $n = n(\lambda)$ (n-PIR) is an interactive protocol between a PPT server S holding a string $z \in \{0,1\}^n$ and a PPT client C holding an index $i \leq n$ which satisfies the following properties:*

- ***Correctness:*** *there exists a negligible function μ such that for every $\lambda \in \mathbb{N}$, $z \in \{0,1\}^n$, $i \in [n]$:*

$$\Pr[\mathsf{Out}_C(\lambda, z, i) = z_i] \geq 1 - \mu(\lambda).$$

- ***Security:*** *there exists a negligible function μ such that for every PPT adversary $\mathcal{A}$, large enough $\lambda \in \mathbb{N}$, $(i, j) \in [n]^2$, and $z \in \{0,1\}^n$:*

$$|\Pr[\mathcal{A}(1^{\lambda+n}, \mathsf{View}_S(\lambda, z, i)) = 1] - \Pr[\mathcal{A}(1^{\lambda+n}, \mathsf{View}_S(\lambda, z, j)) = 1]| \leq \mu(\lambda, n).$$

- ***Efficiency:*** *A PIR is* polylogarithmic *if its communication complexity $c(\lambda, n)$, measured as the worst-case number of bits exchanged between S and C (over their inputs (z, i) and their random coins), satisfies $c(\lambda, n) = \mathsf{poly}(\lambda, \log n)$.*

In this section, we prove the following result:

Main Theorem 2. *Assuming the hardness of the computational Diffie-Hellman assumption against $\mathsf{poly}(n)$-time adversaries, there exists a polylogarithmic n-PIR protocol, with polylogarithmic client computation, and $O(\log n)$ rounds.*

4.1 Laconic Private Set Intersection

Definition 15 (Laconic PSI [ABD+21]). *An ℓPSI scheme $\mathsf{LPSI} = (\mathsf{Setup}, \mathsf{R}_1, \mathsf{S}, \mathsf{R}_2)$ is defined as follows:*

- $\mathsf{Setup}(1^\lambda)$*: Take as input a security parameter 1^λ and outputs a common reference string* crs.
- $\mathsf{R}_1(\mathsf{crs}, S_R)$*: takes as input a* crs *and a receiver set S_R. Outputs a first PSI message* psi_1 *and a state* st.
- $\mathsf{S}(\mathsf{crs}, S_S, \mathsf{psi}_1)$*: takes as input a* crs*, a sender set S_S, and a first PSI message* psi_1*. Outputs a second PSI message* psi_2.
- $\mathsf{R}_2(\mathsf{crs}, \mathsf{st}, \mathsf{psi}_2)$*: takes as input a* crs*, a state* st*, and a second PSI message* psi_2*. Outputs a set $\mathcal{X}$.*

An ℓPSI protocol satisfies the following properties:

– *Correctness: for every sets* (S_R, S_S)*, given* $\mathsf{crs} \xleftarrow{\$} \mathsf{Setup}(1^\lambda)$*,* $(\mathsf{psi}_1, \mathsf{st}) \xleftarrow{\$} \mathsf{R}_1(\mathsf{crs}, S_R)$*,* $\mathsf{psi}_2 \xleftarrow{\$} \mathsf{S}(\mathsf{crs}, S_S, \mathsf{psi}_1)$*, and* $\mathcal{X} \xleftarrow{\$} \mathsf{R}_2(\mathsf{crs}, \mathsf{st}, \mathsf{psi}_2)$*, it holds that* $\mathcal{X} = S_R \cap S_S$ *with probability 1.*
– *Security: the two-round protocol defined by* $\mathsf{LPSI} = (\mathsf{Setup}, \mathsf{R}_1, \mathsf{S}, \mathsf{R}_2)$ *implements the PSI functionality given on Figure 5 in the semi-honest model.*
– *Efficiency: there exists a fixed polynomial* poly *such that both the length of* psi_1 *and the running time of* S *are bounded by* $\mathsf{poly}(\lambda, \log |S_R|)$*.*

Functionality $\mathcal{F}_{\mathsf{psi}}$

Parameters: The PSI functionality $\mathcal{F}_{\mathsf{psi}}$ is parameterised with a universe $\mathcal{U}$.
Setup Phase: The functionality waits until it receives S_R with $S_R \subseteq \mathcal{U}$ from R. Ignores subsequent messages from R.
Send Phase: The functionality waits until it receives S_S with $S_S \subseteq \mathcal{U}$ from S. Sends $S_R \cap S_S$ to R. Ignores subsequent messages from R.

Fig. 5: PSI functionality $\mathcal{F}_{\mathsf{psi}}$

Lemma 16 (ℓPSI from CDH [ABD+21]). *Assuming the security of the computational Diffie-Hellman assumption against* $\mathsf{poly}(n)$*-time adversaries, there exists an ℓPSI protocol for receiver sets of size n with statistical receiver security and computational (semi-honest) sender security.*

4.2 From Laconic PSI to Half-PIR

We define the notion of half-PIR, first introduced in [BIP18] (under the name $\mathsf{Rand}\frac{1}{2}\mathsf{PIR}$). Informally, a half-PIR behaves as a regular PIR with probability 1/2; otherwise, correctness and security might not hold. The receiver gets notified when the scheme successfully worked as intended.

Definition 17. *A half-PIR protocol is defined as an n-PIR (Definition 14) where the correctness and security properties are modified as follows:*

– ***Correctness:*** *there exists a negligible function μ such that for every $\lambda \in \mathbb{N}$, $z \in \{0,1\}^n$, $i \in [n]$:*

$$\Pr[\mathsf{Out}_C(\lambda, z, i) = (z_i, \mathsf{success})] \geq 1/2 - \mu(\lambda).$$

– ***Security:*** *there exists a negligible function μ such that for every PPT adversary $\mathcal{A}$, large enough $\lambda \in \mathbb{N}$, $(i,j) \in [n]^2$, and $z \in \{0,1\}^n$, it holds that $|p_i - p_j| \leq \mu(n, \lambda)$, where for an integer $k \in [n]$, p_k denotes the* conditional *probability* $\Pr[\mathcal{A}(1^{\lambda+n}, \mathsf{View}_S(\lambda, z, k)) = 1 \mid \mathsf{Out}_C(\lambda, z, k)_2 = \mathsf{success}]$.

Below, we recall the definition of pseudorandom functions (PRFs), first introduced in the seminal work of [GGM84]. For simplicity, we restrict our attention to PRFs with key length and output length equal to the security parameter λ.

Definition 18 (Pseudorandom function [GGM84, NR95]). *A* pseudorandom function *with input size m is syntactically defined by a function family $\mathcal{F} = \{F_K : \{0,1\}^{m(\lambda)} \mapsto \{0,1\}^\lambda\}_{\lambda\in\mathbb{N},K\in\{0,1\}^\lambda}$, where the output $F_K(x)$ can be computed from (K, x) in polynomial time, and which satisfies the following security property: for every $\lambda \in \mathbb{N}$ and every oracle PPT attacker $\mathcal{A}$, it holds that*

$$\left|\Pr_K[\mathcal{A}(1^\lambda)^{F_K(\cdot)} = 1] - \Pr_R[\mathcal{A}(1^\lambda)^{R(\cdot)} = 1]\right| \leq \mathrm{negl}(\lambda),$$

*where $K \xleftarrow{\$} \{0,1\}^\lambda$, and $R\colon \{0,1\}^m \mapsto \{0,1\}^\lambda$ is a truly random function. Furthermore, we say that the PRF is $T(\lambda)$-*secure *if the above inequality still holds when $\mathcal{A}$ is additionally given 1^T as input.*

For a high-level intuition of the protocol provided in fig. 6, we refer to the full version of this paper.

Half-PIR from Laconic PSI and PRF.

Parameters: The protocol is parameterised with a security parameter λ, and a database size $n = n(\lambda) \leq 2^\lambda \cdot \mathrm{negl}(\lambda)$. $\{F_K\}_{K\in\{0,1\}^\lambda}$ is a family of $n(\lambda)$-secure PRFs with input size $m = \log n + 1$. The protocol operates in the $\mathcal{F}_{\mathsf{psi}}$-hybrid model, where the universe $\mathcal{U}$ is defined as $\{0,1\}^\lambda$. The server holds an input string $z \in \{0,1\}^n$ and a the client holds an index $i \leq n$.

Protocol: The protocol operates in three steps.

1. The server picks a random PRF key $K \xleftarrow{\$} \{0,1\}^\lambda$ and sends it to the client. The client samples a uniformly random bit $b \xleftarrow{\$} \{0,1\}$, and sets $y \leftarrow F_K(i||b)$.
2. The server constructs the set $S_R = \{F_K(1||z_1), \cdots, F_K(n||z_n)\}$, and queries (sid, S_R) to $\mathcal{F}_{\mathsf{psi}}$, playing the role of the receiver. The client constructs the set $S_S = \{y\}$ and queries S_S to $\mathcal{F}_{\mathsf{psi}}$, playing the role of the sender. The server receives $S_R \cap S_S$.
3. The server indicates whether $S_R \cap S_S$ is empty by sending a bit to the client. If $S_R \cap S_S$ is empty, the client outputs $(1-b, \mathsf{success})$; otherwise, the client outputs (b, fail).

Fig. 6: Half-PIR from Laconic PSI and PRF.

Security analysis. The security analysis is deferred to the full version of this paper.

Instantiating the functionalities. Pseudorandom functions can be constructed from one-way functions [GGM84]. Instantiating the functionality $\mathcal{F}_{\mathsf{psi}}$ with the CDH-based laconic PSI protocol of [ABD+21] involves communication and client

computation $\mathsf{poly}(\lambda, \log |S_R|) = \mathsf{poly}(\lambda, \log n)$ (since $|S_R| = n$). Summing up, we have:

Lemma 19. *Assuming the hardness of the computational Diffie-Hellman assumption against* $\mathsf{poly}(n)$*-time adversaries, there exists a (constant-round) polylogarithmic half-PIR protocol for databases of size* n *(where the client computation is also polylogarithmic).*

4.3 From Polylogarithmic Half-PIR to Polylogarithmic PIR

We now describe a simple generic transformation from Half-PIR to PIR.

Random-index PIR. First, we recall the definition of *random-index* PIR from [GHM+21]:

Definition 20 (Random-Index PIR). *A random-index PIR for database of size* n *is a two-party protocol between a server and a client which implements the random-index PIR functionality given on Figure 7 in the semi-honest model.*

Functionality $\mathcal{F}_{\mathsf{rpir}}$

Parameters: The functionality is parameterised with a database size n.
Server Message: The functionality waits until it receives $z \in \{0,1\}^n$ from the server.
Output: If the client is honest, sample $i \xleftarrow{\$} [n]$ and output (i, z_i) to the client. Otherwise, output z to the client.

Fig. 7: Random-index PIR functionality $\mathcal{F}_{\mathsf{rpir}}$.

Interestingly, random-index PIR was recently shown to imply full-fledged PIR, with only a logarithmic (in n) blowup in communication and rounds, in [GHM+21]:

Lemma 21. *If there exists a random-index PIR protocol for databases of size* n *with communication complexity* $c(\lambda, n)$ *and round complexity* $r(\lambda, n)$*, then there exists an* n*-PIR protocol with communication complexity* $O(c(\lambda, n) \cdot \log n)$ *and round complexity* $O(r(\lambda, n) \cdot \log n)$*.*

We refer to the full version of this paper for a high-level explanation, and to [GHM+21] for a formal proof of Lemma 21.

From half-PIR to random-index PIR. By the above, constructing PIR from half-PIR is reduced to constructing random-index PIR from half-PIR. The latter, however, is straightforward: the client and the server can simply execute a half-PIR, where the client picks its input uniformly at random. At the end of

the protocol, if the client receives fail, both parties simply restart the protocol. By the correctness of the half-PIR, a successful execution will happen after an expected $O(1)$ number of restarts. Below, we describe a slight variant of this where the client runs λ half-PIRs in parallel, and outputs the lexicographically first successful output.

Random-Index PIR from Half-PIR.

Parameters: The protocol is parameterised with a security parameter λ, and a database size $n = n(\lambda) \leq 2^\lambda \cdot \mathrm{negl}(\lambda)$. The server holds an input string $z \in \{0,1\}^n$; the client has no input.

Protocol: The client samples λ uniformly random integers $(i_1, \cdots, i_\lambda) \xleftarrow{\$} [n]^\lambda$. The client and the server run in parallel λ instances of a half-PIR protocol with respective client inputs i_j and server input z. The client receives outputs $\mathsf{Out}_C(\lambda, z, i_j)$.

Output: The client sets j^* to be the lexicographically first j such that $\mathsf{Out}_C(\lambda, z, i_j) = (z_j, \mathsf{success})$ for some bit z_j. The client outputs $(z_{j^*}, \mathsf{success})$. If there is no such j, the client outputs $\perp$ instead.

The security analysis is deferred to the full version of this paper. Combining this protocol with Lemma 21, we get:

Lemma 22. *If there exists a half-PIR protocol for databases of size n with communication complexity $c(\lambda, n)$ and round complexity $r(\lambda, n)$, then there exists an n-PIR protocol with communication complexity $O(\lambda \cdot c(\lambda, n) \cdot \log n)$ and round complexity $O(r(\lambda, n) \cdot \log n)$.*

Putting together Lemmas 19 and 22 finishes the proof of Theorem 2.

References

ABD+21. Navid Alamati, Pedro Branco, Nico Döttling, Sanjam Garg, Mohammad Hajiabadi, and Sihang Pu. Laconic private set intersection and applications. In *Theory of Cryptography Conference*, pages 94–125. Springer, 2021.

ADOS22. Damiano Abram, Ivan Damgård, Claudio Orlandi, and Peter Scholl. An algebraic framework for silent preprocessing with trustless setup and active security. *Cryptology ePrint Archive*, 2022.

AJL+12. Gilad Asharov, Abhishek Jain, Adriana López-Alt, Eran Tromer, Vinod Vaikuntanathan, and Daniel Wichs. Multiparty computation with low communication, computation and interaction via threshold FHE. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 483–501. Springer, Heidelberg, April 2012.

BBDP22. Zvika Brakerski, Pedro Branco, Nico Döttling, and Sihang Pu. Batch-ot with optimal rate. *To appear at Eurocrypt 2022*, 2022.

BCG+19. Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. Efficient pseudorandom correlation generators: Silent OT extension and more. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 489–518. Springer, Heidelberg, August 2019.

BCG+20. Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. Correlated pseudorandom functions from variable-density LPN. In *61st FOCS*, pages 1069–1080. IEEE Computer Society Press, November 2020.

BFKR91. Donald Beaver, Joan Feigenbaum, Joe Kilian, and Phillip Rogaway. Security with low communication overhead. In Alfred J. Menezes and Scott A. Vanstone, editors, *CRYPTO'90*, volume 537 of *LNCS*, pages 62–76. Springer, Heidelberg, August 1991.

BGI16. Elette Boyle, Niv Gilboa, and Yuval Ishai. Breaking the circuit size barrier for secure computation under DDH. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part I*, volume 9814 of *LNCS*, pages 509–539. Springer, Heidelberg, August 2016.

BGI17. Elette Boyle, Niv Gilboa, and Yuval Ishai. Group-based secure computation: Optimizing rounds, communication, and computation. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part II*, volume 10211 of *LNCS*, pages 163–193. Springer, Heidelberg, April / May 2017.

BGW88. Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *20th ACM STOC*, pages 1–10. ACM Press, May 1988.

BI05. Omer Barkol and Yuval Ishai. Secure computation of constant-depth circuits with applications to database search problems. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 395–411. Springer, Heidelberg, August 2005.

BIP18. Elette Boyle, Yuval Ishai, and Antigoni Polychroniadou. Limits of practical sublinear secure computation. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part III*, volume 10993 of *LNCS*, pages 302–332. Springer, Heidelberg, August 2018.

BKS19. Elette Boyle, Lisa Kohl, and Peter Scholl. Homomorphic secret sharing from lattices without FHE. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part II*, volume 11477 of *LNCS*, pages 3–33. Springer, Heidelberg, May 2019.

BLSV18. Zvika Brakerski, Alex Lombardi, Gil Segev, and Vinod Vaikuntanathan. Anonymous IBE, leakage resilience and circular security from new assumptions. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part I*, volume 10820 of *LNCS*, pages 535–564. Springer, Heidelberg, April / May 2018.

CCD88. David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols (extended abstract). In *20th ACM STOC*, pages 11–19. ACM Press, May 1988.

CG97. Benny Chor and Niv Gilboa. Computationally private information retrieval (extended abstract). In *29th ACM STOC*, pages 304–313. ACM Press, May 1997.

CGKS95. Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. Private information retrieval. In *36th FOCS*, pages 41–50. IEEE Computer Society Press, October 1995.

Cha04. Yan-Cheng Chang. Single database private information retrieval with logarithmic communication. In Huaxiong Wang, Josef Pieprzyk, and Vijay Varadharajan, editors, *ACISP 04*, volume 3108 of *LNCS*, pages 50–61. Springer, Heidelberg, July 2004.

CLTV15. Ran Canetti, Huijia Lin, Stefano Tessaro, and Vinod Vaikuntanathan. Obfuscation of probabilistic circuits and applications. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 468–497. Springer, Heidelberg, March 2015.

CM21. Geoffroy Couteau and Pierre Meyer. Breaking the circuit size barrier for secure computation under quasi-polynomial LPN. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part II*, volume 12697 of *LNCS*, pages 842–870. Springer, Heidelberg, October 2021.

CMS99. Christian Cachin, Silvio Micali, and Markus Stadler. Computationally private information retrieval with polylogarithmic communication. In Jacques Stern, editor, *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 402–414. Springer, Heidelberg, May 1999.

Cou19. Geoffroy Couteau. A note on the communication complexity of multiparty computation in the correlated randomness model. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part II*, volume 11477 of *LNCS*, pages 473–503. Springer, Heidelberg, May 2019.

DFH12. Ivan Damgård, Sebastian Faust, and Carmit Hazay. Secure two-party computation with low communication. In Ronald Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 54–74. Springer, Heidelberg, March 2012.

DG17a. Nico Döttling and Sanjam Garg. From selective IBE to full IBE and selective HIBE. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 372–408. Springer, Heidelberg, November 2017.

DG17b. Nico Döttling and Sanjam Garg. Identity-based encryption from the Diffie-Hellman assumption. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 537–569. Springer, Heidelberg, August 2017.

DGI$^+$19. Nico Döttling, Sanjam Garg, Yuval Ishai, Giulio Malavolta, Tamer Mour, and Rafail Ostrovsky. Trapdoor hash functions and their applications. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 3–32. Springer, Heidelberg, August 2019.

DNNR17. Ivan Damgård, Jesper Buus Nielsen, Michael Nielsen, and Samuel Ranellucci. The TinyTable protocol for 2-party secure computation, or: Gate-scrambling revisited. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 167–187. Springer, Heidelberg, August 2017.

FGJI17. Nelly Fazio, Rosario Gennaro, Tahereh Jafarikhah, and William E. Skeith III. Homomorphic secret sharing from paillier encryption. In *Provable Security - 11th International Conference, ProvSec 2017, Xi'an, China, October 23-25, 2017, Proceedings*, volume 10592, pages 381–399, 2017.

Gen09. Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *41st ACM STOC*, pages 169–178. ACM Press, May / June 2009.

GGM84. Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions (extended abstract). In *25th FOCS*, pages 464–479. IEEE Computer Society Press, October 1984.

GHM+21. Craig Gentry, Shai Halevi, Bernardo Magri, Jesper Buus Nielsen, and Sophia Yakoubov. Random-index pir and applications. In *Theory of Cryptography Conference*, pages 32–61. Springer, 2021.

GMW87. Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred Aho, editor, *19th ACM STOC*, pages 218–229. ACM Press, May 1987.

IKM+13. Yuval Ishai, Eyal Kushilevitz, Sigurd Meldgaard, Claudio Orlandi, and Anat Paskin-Cherniavsky. On the power of correlated randomness in secure computation. In Amit Sahai, editor, *TCC 2013*, volume 7785 of *LNCS*, pages 600–620. Springer, Heidelberg, March 2013.

JLS22. Ayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from lpn over $\mathbb{F}_p$, dlin, and prgs in nc^0. *To appear at Eurocrypt 2022*, 2022.

Kil00. Joe Kilian. More general completeness theorems for secure two-party computation. In *32nd ACM STOC*, pages 316–324. ACM Press, May 2000.

KO97. Eyal Kushilevitz and Rafail Ostrovsky. Replication is NOT needed: SINGLE database, computationally-private information retrieval. In *38th FOCS*, pages 364–373. IEEE Computer Society Press, October 1997.

Lip05. Helger Lipmaa. An oblivious transfer protocol with log-squared communication. In Jianying Zhou, Javier Lopez, Robert H. Deng, and Feng Bao, editors, *ISC 2005*, volume 3650 of *LNCS*, pages 314–328. Springer, Heidelberg, September 2005.

NN01. Moni Naor and Kobbi Nissim. Communication preserving protocols for secure function evaluation. In *33rd ACM STOC*, pages 590–599. ACM Press, July 2001.

NR95. Moni Naor and Omer Reingold. Synthesizers and their application to the parallel construction of pseudo-random functions. In *36th FOCS*, pages 170–181. IEEE Computer Society Press, October 1995.

OS07. Rafail Ostrovsky and William E. Skeith III. A survey of single-database private information retrieval: Techniques and applications (invited talk). In Tatsuaki Okamoto and Xiaoyun Wang, editors, *PKC 2007*, volume 4450 of *LNCS*, pages 393–411. Springer, Heidelberg, April 2007.

OSY21. Claudio Orlandi, Peter Scholl, and Sophia Yakoubov. The rise of paillier: Homomorphic secret sharing and public-key silent OT. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part I*, volume 12696 of *LNCS*, pages 678–708. Springer, Heidelberg, October 2021.

RS21. Lawrence Roy and Jaspal Singh. Large message homomorphic secret sharing from DCR and applications. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part III*, volume 12827 of *LNCS*, pages 687–717, Virtual Event, August 2021. Springer, Heidelberg.

Yao86. Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th FOCS*, pages 162–167. IEEE Computer Society Press, October 1986.