Electronic Colloquium on Computational Complexity, Report No. 194 (2026)

Interactive Secret-Key PIR

Nir Bitansky* Geoffroy Couteau† Noam Mazor*

September 18, 2026

Abstract

Private information retrieval (PIR) inherently requires public-key cryptography. A recent line of work suggests that this barrier can be avoided in *secret-key PIR*, where the client first preprocesses an N-bit database and retains only a short secret key. This line of work has yielded communication $\tilde{O}(N^\varepsilon)$ for any constant ε under the Learning Parity with Noise (LPN) assumption in a high-noise regime not known to imply public-key cryptography, and communication $\tilde{O}(\sqrt{N})$ under one-way functions. Whether compression beyond $\sqrt{N}$ can be achieved without relying on *structured assumptions* such as LPN has remained open.

We show that *interaction* enables polylogarithmic communication *in the random oracle model*. We construct a secret-key PIR protocol with $O(\log N)$ rounds and polylogarithmic total communication. Alternatively, for any constant ε, we obtain a constant-round protocol with communication $\tilde{O}(N^\varepsilon)$. We also obtain protocols with similar communication in the plain model under the weakest version of LPN, *with maximal noise rate* $1/2 - o(1)$.

Our main idea, inspired by the free-XOR technique for circuit garbling, is to make secret-key preprocessing homomorphic under XOR, while relying on security against *related-key attacks*.

*New York University. {nbitansky,noammaz}@gmail.com
†CNRS, IRIF, Université Paris Cité, France. couteau@irif.fr

ISSN 1433-8092

Contents

1 Introduction

Private information retrieval (PIR) allows a *client* to read any entry $D[i]$, for $i \in [N]$, from a database $D \in \{0,1\}^N$ stored on a *server*, without revealing i to the server and using only $o(N)$ bits of communication. Ideally, the communication is *polylogarithmic:* $\mathrm{poly}(\lambda, \log N)$ bits, where λ is the security parameter. Since the introduction of PIR [CGKS98; KO97], a central question has been which computational assumptions it requires. An essential barrier has long been known [DMO00]: PIR implies *oblivious transfer* and thus belongs to the domain of *public-key cryptography.*

Aiming to circumvent this barrier, a recent line of work has revisited the question of assumptions in the context of *secret-key PIR* (SK-PIR). In SK-PIR, the client encodes the database D using a short secret key sk in an offline phase, sends the encoded database $\widehat{D}$ to the server, and keeps only the key sk. Later, in the online phase, the client can use sk to make private queries to the database, as in ordinary PIR. The goal is to *minimize the online communication.*

While preprocessing was originally introduced to study possible savings in the server's online work [BIM04], Chen, Ishai, Mour, and Rosen [CIMR26] shift the focus to the model's potential to avoid public-key cryptography; indeed, SK-PIR is not known to imply anything beyond one-way functions [BIPW17]. They construct SK-PIR under the Learning Parity with Noise (LPN) assumption *with a high noise rate*, in a regime where LPN is not known to imply public-key cryptography. Their protocol can achieve communication $N^{\varepsilon} \cdot \mathrm{poly}(\lambda)$ for any constant ε or communication $N^{o(1)}$, assuming super-polynomial hardness.

Bitansky and Mazor [BM26] subsequently show that even one-way functions suffice for significant savings, obtaining communication $\sqrt{N} \cdot \mathrm{poly}(\lambda)$. Their result establishes the feasibility of SK-PIR from minimal assumptions but leaves a substantial gap in communication compared, for instance, with the LPN-based construction of [CIMR26].

> *Is this gap inherent? Could stronger compression be achieved from secret-key primitives? Or does stronger compression inherently require more structure?*

1.1 This Work

We study these questions in the more general model of *interactive* SK-PIR, where, after offline preprocessing, the client and server may interact over multiple rounds during the online phase. In the traditional setting of PIR (without preprocessing), interaction has been shown on several occasions to offer more flexibility in terms of computational assumptions. For instance, Kushilevitz and Ostrovsky [KO00] constructed (slightly sublinear) interactive PIR from trapdoor permutations, and Boyle, Couteau, and Meyer [BCM22] constructed interactive PIR under Computational Diffie-Hellman (CDH). These constructions, too, are within the domain of public-key cryptography.

We show that, for SK-PIR, interaction enables strong compression *in the random oracle model* (ROM), which is traditionally viewed as a *structureless* model and, in particular, is formally separated from public-key cryptography [IR89].

Theorem 1.1 (Informal)**.** *In the ROM, there exists an $O(\log N)$-round SK-PIR protocol with total communication $\mathrm{poly}(\log N, \lambda)$. Alternatively, for any constant $\varepsilon \in (0,1)$, there exists a constant-round protocol with total communication $N^{\varepsilon+o(1)} \cdot \mathrm{poly}(\lambda)$.*

The theorem holds against arbitrary (query-)efficient servers, *including malicious servers.* In the semi-honest setting, the asymptotic parameters are roughly the same, but we obtain better concrete

parameters (e.g., half as many rounds) as well as improved client and server running times, which we discuss below.

The General Theorem: SK-PIR from RKA Security. Our ROM result follows from a more general theorem: SK-PIR protocols with the parameters stated in Theorem 1.1 can be constructed from pseudorandom functions secure against *XOR related-key attacks* ($\oplus$-RKA) [BK03]. These functions remain secure even when the attacker may request evaluations under affinely shifted secret keys $k \oplus s$, for shifts s of its choice. In fact, we can rely on the even weaker primitive of RKA-secure secret-key encryption with pseudorandom ciphertexts.

Theorem 1.2 (Informal, generalization of Theorem 1.1)**.** *Assuming RKA-secure secret-key encryption with pseudorandom ciphertexts, there exists an $O(\log N)$-round SK-PIR protocol with total communication* $\mathrm{poly}(\log N, \lambda)$*. Alternatively, for any constant $\varepsilon \in (0,1)$, there exists a constant-round protocol with total communication $N^{\varepsilon+o(1)} \cdot \mathrm{poly}(\lambda)$.*

While RKA-secure PRFs exist in the random oracle model, relaxing the requirement to encryption yields new results in the plain model. In particular, RKA-secure encryption with pseudorandom ciphertexts can be constructed from *max-noise LPN* [AHI11],[1] namely LPN with noise rate $\frac{1}{2} - \frac{1}{\mathrm{poly}(\lambda)}$. This noise regime is maximal: a negligible gap from $1/2$ would make decoding information-theoretically impossible.

Corollary 1.3. *The PIR protocols stated in Theorem 1.2 can be based on max-noise LPN.*

In contrast, the protocols of [CIMR26] rely on LPN with a much lower noise rate of $k^{-\Omega(1)}$, where k is the LPN dimension. We can achieve polylogarithmic communication under polynomial hardness, without reducing the LPN dimension and assuming super-polynomial hardness. The price we pay is, of course, several rounds of interaction rather than a single round.

Protocol	Communication	Dimension k	Noise rate	Hardness	Rounds	Server storage
[CIMR26]	N^{ε}	N^{ε}	$k^{-\gamma}$, $\gamma < 1$	polynomial	1	$N^{\varepsilon+(1-\varepsilon)/\gamma}$
	$N^{o(1)}$	$N^{o(1)}$		super-polynomial		$N^{1/\gamma+o(1)}$
	$\mathrm{polylog}N$	$\mathrm{polylog}N$		sub-exponential		$N^{1/\gamma+o(1)}$
Our work	N^{ε}	λ	$\frac{1}{2} - \frac{1}{\mathrm{poly}(\lambda)}$	polynomial	$O(1)$	$N^{1+\delta}$
	$\mathrm{polylog}N$				$O(\log N)$	

Table 1: Comparison of our LPN-based protocols with those of [CIMR26]. The stated bounds hold for any constants $\varepsilon, \gamma \in (0,1)$ and $\delta > 0$.

Client and Server Work. Our protocols require substantial online client work. In the malicious setting, the client work can be as large as $\widetilde{\Theta}(N^{1+\delta})$ for polylogarithmic communication and arbitrarily small δ, or as large as $\widetilde{\Theta}(N^{1-\varepsilon/2+o(1)})$ for a constant number of rounds and communication $N^{\varepsilon+o(1)}$. We note that the product of client work and communication is also substantial in previous SK-PIR constructions [CIMR26; BM26]. In the LPN-based construction of [CIMR26], this product is $\widetilde{\Theta}(N^{2/3+\delta})$ for arbitrarily small δ, whereas in the construction of [BM26], based on one-way functions, it is $\widetilde{\Theta}(N)$. The server work in our protocols is $\widetilde{\Theta}(N^{1+\delta})$ for polylogarithmic communication and

[1] In [AHI11], this is stated for a noise rate of $1/2 - \gamma$ with constant γ. However, γ can be made inverse polynomial by using a repetition code rather than a constant-rate code, at the cost of a polynomial increase in ciphertext length.

arbitrarily small δ, and $\widetilde{\Theta}(N^{1+\varepsilon+o(1)})$ for a constant number of rounds and communication $N^{\varepsilon+o(1)}$. In the semi-honest setting, the product of client work and communication, as well as the server running time, can be reduced to $\widetilde{\Theta}(N)$.

Communication	Rounds	Client work	Server work	Parameter
$(\log N)^{O(\delta^{-1})}$	$O(\log N)$	$N^{1+\delta}$	$N^{1+\delta}$	any constant $\delta > 0$
$N^{\eta+o(1)}$	$O(\eta^{-1})$	$N^{1+o(1)}$	$N^{1+o(1)}$	any $\eta = o(1)$
$N^{\varepsilon+o(1)}$	$O(\varepsilon^{-1})$	$N^{1-\varepsilon/2+o(1)}$	$N^{1+\varepsilon+o(1)}$	any constant $\varepsilon \in (0,1)$

Table 2: Achievable trade-offs for maliciously secure SK-PIR from XOR-RKA-secure encryption (Corollary 7.2). All quantities are per query and hide $\mathrm{poly}(\lambda)$ factors. The $o(1)$ terms are $O(\sqrt{\log\log N/\log N})$.

Remark 1.4 (Reducing the server's work)**.** *In the last row of Table 2, the server's work can be reduced from $N^{1+\varepsilon+o(1)}$ to $N^{1+o(1)}$, matching the storage bound (the first two rows already match storage), if the XOR-RKA-secure encryption is replaced by an XOR-RKA-secure PRF (hence in particular in the random-oracle model). This variant leverages the determinism of PRF evaluation to let the server perform a cheaper comparison with its local evaluations in a batch setting. See Remark 4.3 for details.*

1.2 Technical Overview

We now describe the main ideas behind our constructions.

The Starting Point: A Simple Half PIR. The starting point for our construction is a relaxation of PIR known as *Half PIR* (1/2PIR) [BIP18]. A 1/2PIR has the same functionality as a plain PIR, except that in every execution, with probability $\approx 1/2$, the client's secret index leaks. Concretely, upon a query i, the client obtains the database entry $D[i]$ as well as a status $s \in \{\mathsf{succ}, \mathsf{fail}\}$, which indicates whether leakage occurred.

Our first observation is that a *secret-key* 1/2PIR (SK-1/2PIR) can be constructed quite simply from one-way functions. The client's secret key consists of $2N$ keys $\{k_{j,b} : j \in [N], b \in \{0,1\}\}$ for a pseudorandom function (PRF) F. (In the actual construction, these keys will be compressed into one short key k, a step we omit here for simplicity.) To preprocess the database $D \in \{0,1\}^N$, the client sends the server the corresponding keys $\{k_j := k_{j,D[j]}\}_{j\in[N]}$, as well as the database D itself. To later query index i, the client samples a guess g_i for $1 - D[i]$ and a random string x. The client then sends the server x and $y = F(k_{i,g_i}, x)$. The server applies the PRF to compute $y_j = F(k_j, x)$ for *all* keys k_j in its possession. If any y_j matches the client's string y, the server returns fail and the client deduces $D[i] = g_i$; otherwise, the server returns succ and the client deduces $D[i] = 1 - g_i$.

For correctness, provided that the PRF output is sufficiently long, the following holds with overwhelming probability: if $g_i \neq D[i]$, then $k_{i,g_i} \notin \{k_j\}$ and y does not match any y_j; if $g_i = D[i]$, then y matches exactly one string, namely $y_i = F(k_i, x)$. For (half) security, whenever $g_i \neq D[i]$, and provided that the random strings x in different executions are sufficiently long and do not collide, y is pseudorandom from the server's perspective and leaks nothing about i.

For the time being, we assume that the server is *semi-honest*. In particular, the server honestly reports whether a match (and hence leakage) occurred. This is crucial for the security argument above; we will later see how to remove this assumption.

What Is 1/2PIR Good For? While 1/2PIR may seem quite weak, it was shown to imply (in the plain, non-secret-key setting) slightly sublinear PIR [BIP18], as well as *interactive* polylogarithmic PIR [GHMNY21; BCM22]. Let us elaborate on the latter implication, which will be instrumental in our construction.

As observed in [BCM22], 1/2PIR gives rise to another relaxation of PIR known as Random-Index PIR (RPIR) [GHMNY21]. In RPIR, the client cannot obtain a database entry of their choice, but instead obtains a *randomly distributed index* together with the corresponding entry, namely $(u, D[u])$; the server, on the other hand, learns nothing about the index u. To obtain RPIR from 1/2PIR, one can invoke 1/2PIR sufficiently many times in parallel, each time with a fresh random index, and use the first non-leaky execution. Since leakage depends only on the client's independent guesses, the probability of error decays exponentially with the number of parallel instances.

RPIR, in turn, was shown in [GHMNY21] to yield an interactive PIR scheme in which the client can obtain any entry of their choice after several rounds of interaction with the server. The basic idea is to use RPIR to recursively fold the database. A client interested in obtaining $D[i]$ first invokes RPIR to obtain a random entry $(u, D[u])$ and sends $f = i \oplus u$ to the server; here we assume $N = 2^n$ and interpret the indices as strings in $\{0,1\}^n$.

The server will then create a *folded* database:

$$D_f[j] = D[j] \oplus D[j \oplus f].$$

The client's next goal is to learn $D_f[u] = D[u] \oplus D[i]$, from which they can recover $D[i]$ using their knowledge of $D[u]$. The savings come from the fact that f (publicly) partitions the indices $[N]$ into pairs $(j, j \oplus f)$ on which D_f takes the same value. Thus, we have effectively reduced the size of the database by a factor of two. To obtain polylogarithmic PIR, we apply this procedure recursively over n rounds.

Back to Secret-Key 1/2PIR. Can the same approach be applied if we start from an SK-1/2PIR? The transformation from 1/2PIR to RPIR works just as well in the secret-key setting. However, the folding idea, which recursively uses RPIR to obtain polylogarithmic interactive PIR, seems to be at odds with preprocessing. The folded database D_f is determined during the online phase and depends on the client's index i, so its computation cannot simply be moved to the preprocessing phase.

Homomorphic Preprocessing. The basic idea behind our construction is to make database preprocessing homomorphic with respect to these folding operations. Roughly speaking, the server, given a preprocessed version $\widehat{D}$ of the database D and the folding index f, should be able to obtain a preprocessed version $\widehat{D}_f$ of the folded database D_f. The client, in turn, given the secret key sk and f, should be able to obtain a new secret key sk_f that is compatible with $\widehat{D}_f$. Furthermore, we wish to support a sequence of folding operations $f_1, \ldots, f_n$ to allow for the recursive reduction above. Crucially, these operations should be *local*, with little or no extra interaction between the client and the server.

Our solution is inspired by the *free-XOR* technique for garbled circuits [KS08]. For each database index i, rather than sampling a pair of keys $(k_{i,0}, k_{i,1})$, we sample only $k_{i,0}$ at random and set $k_{i,1} = k_{i,0} \oplus \Delta$ for a global random offset Δ, used across all indices i. As before, the server obtains a single key $k_{i,D[i]}$ per entry, according to the database. In particular, the offset Δ remains secret. This global offset allows *free XORing of keys:*

$$k_{i,b_i} \oplus k_{j,b_j} = k_{i,0} \oplus k_{j,0} \oplus (b_i \oplus b_j) \cdot \Delta.$$

Defining new virtual keys

$$k_{ij,0} = k_{i,0} \oplus k_{j,0}, \qquad k_{ij,1} = k_{ij,0} \oplus \Delta,$$

we have

$$k_{i,b_i} \oplus k_{j,b_j} = k_{ij,b_i \oplus b_j}.$$

This, in particular, supports the folding described above. The keys for position i in the folded database D_f are:

$$k^f_{i,0} = k_{i,0} \oplus k_{i \oplus f,0}, \qquad k^f_{i,1} = k^f_{i,0} \oplus \Delta,$$

and by XORing its keys

$$k_{i,D[i]} \oplus k_{i \oplus f, D[i \oplus f]},$$

the server obtains:

$$k^f_{i,D[i] \oplus D[i \oplus f]} = k^f_{i,D_f[i]}.$$

This allows us to apply the same recursion as before to obtain SK-PIR with polylogarithmic communication. While the total communication is only polylogarithmic in N, the client's work is $N \cdot \mathrm{poly}(\log N, \lambda)$: computing a folded key in round r requires XORing two folded keys from the previous round $r-1$. The server's work, which consists of folding the database and scanning for a match polylogarithmically many times, is similarly $N \cdot \mathrm{poly}(\log N, \lambda)$.

In the body, we consider a generalization of the above solution, allowing the client to query the virtual database MD for any binary matrix M, known to both the client and server. This captures the above folding as well as other useful forms of folding that we touch on later.

Related-Key Attacks and Security. The security of the scheme boils down to the security of the corresponding SK-1/2PIR when applied to a folded database $D_{\boldsymbol{f}}$ in round r, where $\boldsymbol{f} = (f_1, \ldots, f_r)$, with the client's folded secret keys $\{k^{\boldsymbol{f}}_{i,0}, k^{\boldsymbol{f}}_{i,1}\}$. The invariant throughout all rounds is that when the server receives a value $y = F(k, x)$ as part of the 1/2PIR protocol, the corresponding key k has one of two forms. It may be a known linear combination (corresponding to $\boldsymbol{f}$) of the original server keys, which results in a leakage event. Alternatively, $k = k' \oplus \Delta$, where k' is likewise a known linear combination of the original server keys. We would like to argue that, in the second case, y is pseudorandom, just as in the previous analysis.

This is exactly the setting of (affine) *related-key attacks* (RKA) [BK03]. We may view the hidden random offset Δ as the secret key and require pseudorandomness even when evaluating the corresponding PRF under related keys $\Delta \oplus k'$, for known affine shifts k'. Similar related-key attacks also arise in free-XOR garbling [KS08; App13].[2]

Instantiating F with a random oracle gives the required RKA guarantees [KS08]. For our plain-model result from max-noise LPN, we need to generalize the construction slightly, since an RKA-secure PRF is not known to follow from LPN. Instead, we use an RKA-secure secret-key encryption scheme with pseudorandom ciphertexts, which follows from max-noise LPN [AHI11]. We encrypt a random x, and check for a match when decrypting. For simplicity, throughout this introduction, the reader may continue to think of the PRF-based solution described thus far.

[2] Free-XOR garbling also deals with key-dependent messages, a concern which does not arise in our setting.

Malicious Servers. So far, we have assumed that the server is semi-honest. Specifically, in our $1/2$PIR, we assumed that the server honestly reports whether leakage occurred, namely, whether there was a match $y = y_j$. In the construction of RPIR from $1/2$PIR, reliably learning whether leakage occurred was crucial to ensuring that the random index u remained hidden from the server. Indeed, if u leaks, then using it for folding would reveal the client's query index $i = f \oplus u$.

One generic solution, in the plain model, is to use *succinct arguments of knowledge* [Kil92; Mic00]. The client would hash the preprocessed database and maintain the resulting short digest d; then, after every round, the server could give a succinct proof of knowledge that it acted consistently with a preprocessed database matching the digest d and the transcript so far. This solution, however, has several downsides. It does not make blackbox use of the underlying cryptographic primitives; in particular, it does not relativize and thus cannot be applied in the random oracle model. In the plain model, it requires the additional assumption of (multi-)collision-resistant hashing, which is not known to be implied by XOR-RKA encryption or max-noise LPN.

We give a more combinatorial solution that applies in both the random oracle model and the plain model without requiring additional assumptions. The solution has two conceptual steps: (1) obtain an interactive PIR protocol that withstands malicious servers except with some vanishing but non-negligible probability; (2) amplify security. Throughout both stages, we wish to preserve the polylogarithmic communication complexity.

For the first step, let us go back to the construction of RPIR from $1/2$PIR. Recall that here the client and server run t parallel copies of $1/2$PIR, with independent random indices $u_1, \ldots, u_t$. Leakage occurs in an attempt $j \in [t]$ if the client incorrectly guesses the value $1 - D[u_j]$, in which case y will match y_{u_j}. The worry is that such a match might occur without the server reporting it to the client. While there does not seem to be an immediate way for the server to prove that leakage did not occur, there is a short proof that a match $y = y_{u_j}$ (and hence leakage) did occur: the proof is simply the corresponding key k_{u_j}. We use this to our advantage.

We will check the statistics of leakage events, demanding that whenever leakage occurs, the server provides proof of this fact. By concentration, we know that with overwhelming probability the number of occurrences deviates from the mean $t/2$ by at most $t^{0.51}$. Accordingly, we can insist that the server exhibits $t/2 - t^{0.51}$ proven leakage events, and, assuming it does, we can rest assured that at least $t/2 - t^{0.51}$ of the remaining $t/2 + t^{0.51}$ indices are safe to use. Thus, if we sample one of the remaining indices at random, it will be safe except with probability $\delta \approx t^{-0.49}$. Choosing a polylogarithmic t will be sufficient for our needs (in the body, we perform a tighter analysis).

Once equipped with a PIR protocol such as the one above, we amplify its security using an amplifier from [BGIK22], originally used in the related context of distributed point functions. Roughly speaking, the amplifier encodes the database using an appropriate locally decodable code (e.g., a Reed–Muller code) and performs randomized local decoding. The code has the property that the underlying entry $D[i]$ can be decoded by reading a polylogarithmic number q of codeword locations. Furthermore, any $\approx \delta$ fraction of these q locations are uniformly random and independent of i. By reading each of these q locations with the δ-erroneous PIR described above, we guarantee that sufficiently few locations are leaked and that the leaked locations reveal nothing about i.

We note that amplifying computational security tends to be tricky and often involves delicate coupling arguments or the use of the hardcore lemma (cf. the discussion in [ABG26]).[3] Our setting is simpler than the one in [BGIK22], as a faulty execution can be efficiently detected given the database and secret key. This allows us to avoid the common difficulties of computational amplification and

[3] In particular, a gap was identified [GZSP25] in the amplification proof of [BGIK22].

reduce directly to known parallel repetition theorems for interactive proofs (cf. [Gol25]).

A side effect of our amplification step is an increase in both the client and server running times from $\widetilde{\Theta}(N)$ to $\widetilde{\Theta}(N^{1+\delta})$ for an arbitrarily small constant δ. This is due to the code blowup in the polylogarithmic parameter regime.

Communication vs. Rounds. So far, we have focused on polylogarithmic PIR using the folding technique from [GHMNY21]. The resulting PIR has $\log(N)$ rounds of interaction with $\mathrm{poly}(\log(N), \lambda)$ communication in each round. To reduce the number of rounds, we give a generalized folding technique that trades off total communication against the number of rounds. Focusing on constant-round protocols, we show that for any constant ε there exists a protocol with $N^{\varepsilon+o(1)} \cdot \mathrm{poly}(\lambda)$ communication and a constant number of rounds (depending on ε^{-1}).[4]

The basic idea is to partition the indices $[N]$ into $\ell \approx N^{\varepsilon}$ consecutive blocks of size $b = N/\ell$. It will be useful to view the database D as an $\ell \times b$ matrix, with the blocks as its rows. The client is interested in learning a specific entry $D[i^*, j^*]$. For each row $i \in [\ell]$, we run the RPIR protocol to learn $(u_i, D[i, u_i])$ for a random index $u_i \in [b]$ in that row. The client then chooses a random $\tau \leftarrow \mathbb{Z}_b$ and sends the ℓ random shifts

$$\boldsymbol{s} = (\tau - u_1, \ldots, \tau - u_{i^*-1}, \tau - j^*, \tau - u_{i^*+1}, \ldots, \tau - u_\ell),$$

where all computations are modulo b. We then perform a cyclic shift of each row by the prescribed shift, as illustrated in Figure 1. This has the effect of aligning the known entries

$$D[1, u_1], \ldots, D[i^* - 1, u_{i^*-1}], D[i^* + 1, u_{i^*+1}], \ldots, D[\ell, u_\ell]$$

with the desired entry $D[i^*, j^*]$, so that they all appear in the random column τ. The server then computes the folded database $D_{\boldsymbol{s}}$ by XORing the rows, and the protocol is executed recursively on the smaller database.

In the semi-honest setting, for communication $N^{\varepsilon} \cdot \mathrm{poly}(\lambda)$, we obtain ε^{-1} rounds, client work $N^{1-\varepsilon} \cdot \mathrm{poly}(\lambda)$, and server work $N \cdot \mathrm{poly}(\lambda)$. In the malicious setting, we must account for the overheads of amplification. For communication $N^{\varepsilon+o(1)} \cdot \mathrm{poly}(\lambda)$, we obtain $2\varepsilon^{-1}$ rounds, client work $N^{1-\varepsilon/2+o(1)} \cdot \mathrm{poly}(\lambda)$, and server work $N^{1+\varepsilon+o(1)} \cdot \mathrm{poly}(\lambda)$.

[4] A generalized folding technique using random partitions into larger sets (rather than pairs), considered in [GHMNY21], could potentially improve our parameters but leads to client communication of $\approx N$. The folding technique that we introduce here is efficient and can also be used in [GHMNY21] (regardless of preprocessing).

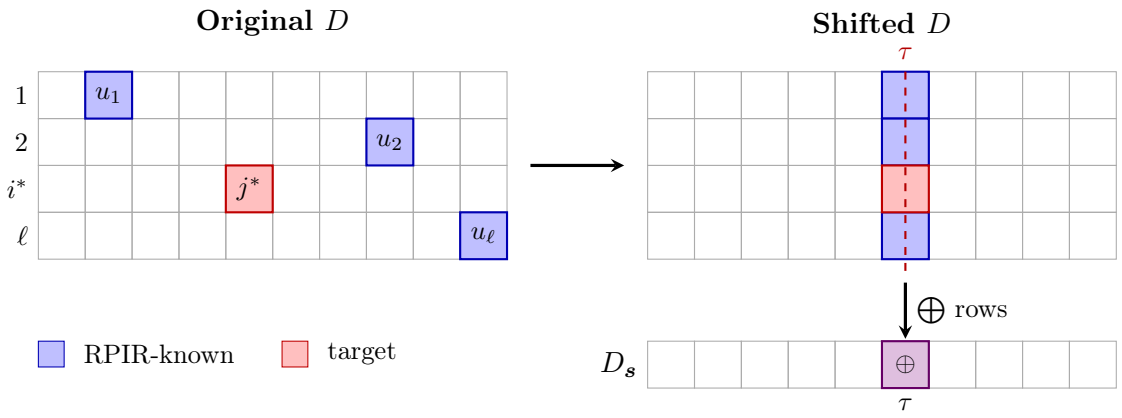

Figure 1: Generalized folding. The client cyclically shifts each row so that the entries learned through RPIR (blue) and the target entry (red) align at the random column τ. XORing the shifted rows produces $D_{\boldsymbol{s}}$, whose τ-th entry is the target bit XORed with bits already known to the client.

1.3 More Related Work

Multi-Server and Single-Server PIR. PIR was introduced by Chor, Goldreich, Kushilevitz, and Sudan [CGKS98] in the multi-server setting, where the database is replicated across non-colluding servers. This setting admits information-theoretic privacy with sublinear communication and is closely connected to locally decodable codes [KT00]. Kushilevitz and Ostrovsky [KO97] subsequently showed that replication can be avoided under computational assumptions, introducing single-server computational PIR. Unlike multi-server PIR, single-server PIR without preprocessing inherently requires public-key cryptography [DMO00]. Our focus is on how secret-key preprocessing and interaction change the assumptions needed in this single-server setting.

Preprocessing for Online Server Efficiency. In the standard PIR model, answering a query requires essentially a full scan of the database. Beimel, Ishai, and Malkin [BIM04] introduced PIR with preprocessing to reduce this online server work: the database is encoded once in an offline phase, after which queries can be answered by accessing only a small part of the encoding. The goal of achieving both sublinear communication and sublinear online server work is commonly referred to as *doubly efficient PIR*. Boyle, Ishai, Pass, and Wootters [BIPW17] and Canetti, Holmgren, and Richelson [CHR17] gave candidate single-server constructions with secret-key preprocessing, achieving polylogarithmic communication and online server work under assumptions concerning secretly permuted Reed–Muller codes. Lin, Mook, and Wichs [LMW23] later constructed doubly efficient PIR from Ring LWE with public preprocessing, so that the database encoding does not depend on a client's secret key.

The works of Chen, Ishai, Mour, and Rosen [CIMR26] and Bitansky and Mazor [BM26], as well as ours, revisit secret-key preprocessing with a different emphasis: minimizing communication under weaker computational assumptions, without requiring sublinear online server work. Thus, our polylogarithmic-communication protocols do not claim the online server efficiency of doubly efficient PIR. There are also preprocessing models in which the client maintains and updates database-dependent state, as well as the related ORAM model, in which the server can be stateful too. These differ from our setting, where the same encoded database and short secret key are reused without

updates between queries. We refer the reader to the discussion in [CIMR26, Section 3.1] for a broader account of preprocessing models, their efficiency goals, and their relation to ORAM.

Recent Improvements in Online Efficiency. Chen, Ishai, Jain, Mour, Rosen, and Xing [Che+26] give candidate doubly efficient SK-PIR schemes with $N^{o(1)}$ communication and online server work, together with constant multiplicative storage overhead that approaches one in suitable parameter regimes. Their constructions build on the permuted-code approach of [BIPW17; CHR17], using new locally decodable codes with improved rate, locality, and smoothness trade-offs. Security relies on assumptions of the same structured type as those underlying the earlier permuted-code constructions. Their focus on reducing online server work and storage overhead is complementary to ours: we use interaction to minimize communication under generic RKA security, with instantiations in the ROM and from max-noise LPN.

Encrypted Matrix–Vector Products. Benhamouda, Chen, Halevi, Ishai, Krawczyk, Mour, Rabin, and Rosen [Ben+25] extend the secret-code approach to encrypted matrix–vector products. A client uploads an encrypted matrix while retaining only a short secret key, and later issues encrypted vector queries so that the parties obtain compact shares of the product, hiding both the matrix and the query from the server. Their field-agnostic constructions use secret dual linear codes and LPN- or LSN-type assumptions, and aim for communication and computation close to the corresponding cleartext costs. This supports applications beyond private retrieval, including encrypted search and secure machine learning. The connection to our work is the use of reusable secret-key preprocessing compatible with linear operations; our construction instead uses XOR-homomorphic preprocessing and RKA security to support recursive database folding.

1.4 Open Questions

Overall, there is still a gap in our understanding of SK-PIR.

Improvements in Efficiency or Assumptions. The ultimate goal is a single-round SK-PIR protocol with polylogarithmic communication and online client work, based only on one-way functions. Short of achieving all these goals simultaneously, new trade-offs among communication, round complexity, client work, and computational assumptions would already be interesting.

Barriers and Locally Decodable Codes. It is also natural to seek separations between SK-PIR with strong efficiency guarantees and one-way functions, or even random oracles. We note, however, that progress in this direction is constrained by our limited understanding of locally decodable codes (LDCs). In particular, starting from secret-key $1/2$PIR, which we build based on one-way functions, a q-query smooth LDC can be used to obtain a q-round SK-PIR protocol with communication and online client work roughly 2^q, via a transformation similar to that of [BIP18].

For example, an LDC encoding N bits into a block of length $\mathrm{poly}(N)$ with $q = \varepsilon \log N$ queries would yield SK-PIR from one-way functions with $O(\log N)$ rounds and both communication and online client work $\widetilde{O}(N^{\varepsilon})$. Thus, ruling out these SK-PIR parameters from one-way functions, or in the ROM, would also rule out such LDCs. This connection poses a challenge for proving barriers to simultaneously reducing communication and client work when interaction is allowed.

2 Preliminaries

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

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

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

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

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

We will use the following definition of interactive oracle.

Definition 2.1 (Interactive oracle)**.** *Let Π be an interactive algorithm defined by a (possibly randomized) next-message function $\Pi(x, \rho, \tau)$ mapping an input x, a random tape ρ, and a transcript τ (the ordered list of messages sent and received so far in a session) to either a next message m or a local output* out *(which terminates the session). The* interactive oracle induced by Π with common input w*, denoted $\mathcal{O}_\Pi(w, \cdot)$, is the following stateful oracle. It maintains a set of sessions, each consisting of a session identifier* sid*, a session input x_{sid}, a random tape ρ_{sid}, and a transcript τ_{sid}. An oracle-aided algorithm $\mathcal{A}$ may adaptively issue two types of queries:*

- (new, x)*: the oracle creates a new session with a fresh identifier* sid*, sets $x_{\mathsf{sid}} := (w, x)$, samples a uniformly random tape ρ_{sid}* independently of all other sessions*, initializes τ_{sid} to the empty transcript, and returns* sid *together with Π's first message $\Pi(x_{\mathsf{sid}}, \rho_{\mathsf{sid}}, \tau_{\mathsf{sid}})$ if Π speaks first (appending it to τ_{sid}).*
- $(\mathsf{msg}, \mathsf{sid}, m)$ *for an active session* sid*: the oracle appends m to τ_{sid}, computes $\Pi(x_{\mathsf{sid}}, \rho_{\mathsf{sid}}, \tau_{\mathsf{sid}})$, appends the resulting message to τ_{sid} and returns it to $\mathcal{A}$. If Π instead terminates with a local output, the session is closed and $\mathcal{A}$ receives only a fixed termination symbol* done *(in particular, local outputs are* not *given to $\mathcal{A}$ unless the definition using the oracle says so explicitly).*

Each session's transcript is append-only: $\mathcal{A}$ cannot rewind, reset, fork, or re-run a session on a modified message, and sessions share no state beyond the common input w. We stress that $\mathcal{A}$ may run arbitrarily many sessions concurrently and interleave its queries adaptively.

2.1 Secret-Key Interactive PIR

We define interactive secret-key PIR.

Definition 2.2 (Secret-key Interactive PIR)**.** *A* secret-key interactive private information retrieval scheme (sk-iPIR) *consists of PPT algorithms* $(\mathsf{Gen}, \mathsf{Preprocess})$ *and a two-party protocol* $(\mathsf{Client}, \mathsf{Server})$ *with the following syntax:*

- $\mathsf{Gen}(1^\lambda, 1^N) \to \mathsf{sk}$*: on input a security parameter λ and a database size N, outputs a secret key* sk*.*

- $\mathsf{Preprocess}(\mathsf{sk}, D) \to \widehat{D}$*: on input* sk *and a database* $D \in \{0,1\}^N$*, outputs an encoding* $\widehat{D}$*. We require, w.l.o.g. for this paper, that* D *is part of* $\widehat{D}$ *(the encoding hides nothing about the data from the server; only queries are private).*
- $\langle \mathsf{Client}(\mathsf{sk}, i), \mathsf{Server}(\widehat{D}) \rangle$*: an interactive protocol in which the client holds* (sk, i) *for* $i \in [N]$*, the server holds* $\widehat{D}$*, and at the end the client halts with a local output* $b \in \{0,1\}$*.*

Correctness. *For any* $N = \mathrm{poly}(\lambda)$ *there is a negligible function* μ *such that the following holds. For every* $\lambda \in \mathbb{N}$*,* $D \in \{0,1\}^N$ *and* $i \in [N]$*,*

$$\Pr\Big[\big(\langle \mathsf{Client}(\mathsf{sk}, i), \mathsf{Server}(\widehat{D}) \rangle\big)_{\mathsf{Client}} = D[i]\Big] \geq 1 - \mu(\lambda),$$

where the probability is over $\mathsf{sk} \leftarrow \mathsf{Gen}(1^\lambda, 1^N)$*,* $\widehat{D} \leftarrow \mathsf{Preprocess}(\mathsf{sk}, D)$ *and the coins of both parties.*

Round and communication complexity. *The protocol has* round complexity $R = R(N, \lambda)$ *if in every execution at most* $2R$ *messages are exchanged (we count a round as one client message followed by one server message), and* communication complexity $c = c(N, \lambda)$ *if the total length of all exchanged messages is at most* $c(N, \lambda)$ *with probability* 1*. We call the scheme* fully succinct *if* $c(N, \lambda) \leq \mathrm{poly}(\lambda, \log N)$*.*

Definition 2.3 (Malicious Security)**.** *A* sk*-iPIR is* maliciously secure *if for every oracle-aided non-uniform PPT* $\mathcal{A}$ *and every polynomial* $N = N(\lambda)$ *there is a negligible* μ *such that for every* $\lambda \in \mathbb{N}$ *and every* $D \in \{0,1\}^{N(\lambda)}$*:*

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

where $\mathsf{sk} \leftarrow \mathsf{Gen}(1^\lambda, 1^N)$*,* $\widehat{D} \leftarrow \mathsf{Preprocess}(\mathsf{sk}, D)$*,* $\mathcal{O}_{\mathsf{Client}}(\mathsf{sk}, \cdot)$ *is the interactive oracle that, on session input* $i \in [N]$*, runs the honest client* $\mathsf{Client}(\mathsf{sk}, i)$ *(with* $\mathcal{A}$ *playing the server, fresh independent coins per session, no rewinding, local output withheld), and* $\mathcal{O}'_{\mathsf{Client}}(\mathsf{sk}, \cdot)$ *is the interactive oracle that ignores the session input and runs* $\mathsf{Client}(\mathsf{sk}, 1)$*.*

Definition 2.4 (Semi-honest Security)**.** *A* sk*-iPIR is* semi-honestly secure *if, with the same quantifiers, the above holds with the interactive oracles replaced by the (non-interactive) oracles* $V(\mathsf{sk}, \widehat{D}, \cdot)$ *and* $V'(\mathsf{sk}, \widehat{D}, \cdot)$*, where* $V(\mathsf{sk}, \widehat{D}, i)$ *outputs a fresh sample of the view of the honest server (its input* $\widehat{D}$*, its random coins, and the transcript) in* $\langle \mathsf{Client}(\mathsf{sk}, i), \mathsf{Server}(\widehat{D}) \rangle$*, and* $V'(\mathsf{sk}, \widehat{D}, i) := V(\mathsf{sk}, \widehat{D}, 1)$*.*

Remark 2.5. *Malicious security implies semi-honest security: an adversary in the semi-honest game can be emulated by one in the malicious game that runs the honest server code itself.*

2.2 RKA-Secure Encryption

Definition 2.6 (RKA-secure encryption with pseudorandom ciphertexts)**.** *An* $\mathsf{encryption\ scheme}$ $\mathsf{with\ pseudorandom\ ciphertexts}$ *consists of PPT algorithms* $(\mathsf{Enc}, \mathsf{Dec})$ *such that:*

- $\mathsf{Enc}(\mathsf{sk}, m)$*: The encryption algorithm* $\mathsf{Enc}\colon \{0,1\}^\lambda \times \{0,1\}^\lambda \to \{0,1\}^{\ell(\lambda)}$*. Takes a secret key* $\mathsf{sk} \in \{0,1\}^\lambda$ *and a message* $m \in \{0,1\}^\lambda$*, and outputs a ciphertext* $c \in \{0,1\}^{\ell(\lambda)}$*.*
- $\mathsf{Dec}(\mathsf{sk}, c)$*: The decryption algorithm. Takes a secret key* $\mathsf{sk} \in \{0,1\}^\lambda$ *and a ciphertext* c*, and outputs a message* m*.*

Correctness: *For any* $m \in \{0,1\}^\lambda$,

$$\Pr[\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{sk}, m)) = m] \geq 1 - \mathrm{neg}(\lambda)$$

where $\mathsf{sk} \leftarrow \{0,1\}^\lambda$.

Security: *Let* Φ *be a family of related key deriving functions.* $(\mathsf{Enc}, \mathsf{Dec})$ *is* secure against Φ-restricted Related Key Attacks (Φ-RKA-secure) *if for every oracle-aided non-uniform PPT* $\mathcal{A}$ *there exists a negligible function* μ*, such that for any* λ*,*

$$\left|\Pr\left[\mathcal{A}^{\mathsf{Enc}(RK(\mathsf{sk},\cdot),\cdot)}(1^\lambda) = 1\right] - \Pr\left[\mathcal{A}^{\Pi}(1^\lambda) = 1\right]\right| \leq \mu(\lambda),$$

where $\mathsf{sk} \leftarrow \{0,1\}^\lambda$*,* $RK(\mathsf{sk}, \phi) = \phi(\mathsf{sk})$ *for every* $\phi \in \Phi$*, and* Π *is an oracle that given* $\phi \in \Phi$ *and a message* m *outputs a uniformly random string of length* $\ell(|\mathsf{sk}|)$*, sampled independently on each oracle call (even for repeated queries).*

We say that $(\mathsf{Enc}, \mathsf{Dec})$ is XOR-RKA-secure if it is Φ-RKA secure with respect to the function family $\Phi = \left\{\phi_\Delta(\mathsf{sk}) = \mathsf{sk} \oplus \Delta \colon \Delta \in \{0,1\}^\lambda\right\}$.

Remark 2.7 (Pseudorandomness and anonymity)**.** *We require pseudorandom ciphertexts as a convenient way to ensure* key anonymity*: ciphertexts should not reveal which of the hidden related keys was used, even when the plaintext is known. Pseudorandomness is stronger than anonymity, and we use the stronger formulation throughout for simplicity.*

Below we give two instantiations of RKA-secure encryption with pseudorandom ciphertexts.

ROM-based construction. The first, implicitly given in [KS08] and proven here for completeness, is in the Random Oracle Model. Let $H \colon \{0,1\}^\lambda \times \{0,1\}^\lambda \to \{0,1\}^\lambda$ be a random oracle. Define

$$\mathsf{Enc}^H(k, m) := (\rho,\, H(k, \rho) \oplus m) \text{ for } \rho \leftarrow \{0,1\}^\lambda, \qquad \mathsf{Dec}^H\big(k, (\rho, y)\big) := H(k, \rho) \oplus y,$$

with ciphertext length $\ell_{\mathsf{Enc}}(\lambda) = 2\lambda$.

Theorem 2.8 (ROM instantiation)**.** *In the random-oracle model, the scheme above has perfect correctness XOR-RKA pseudorandom ciphertexts: for every adversary making at most* q_H *direct* H*-queries and* q_E *encryption-oracle queries,*

$$\left|\Pr\left[\mathcal{A}^{\text{real}} = 1\right] - \Pr\left[\mathcal{A}^{\text{ideal}} = 1\right]\right| \leq \frac{q_E^2}{2^{\lambda+1}} + \frac{q_H\, q_E}{2^\lambda}.$$

Proof. Lazily sample H. The real oracle answers the j-th query (ϕ_{Δ_j}, m_j) with $(\rho_j,\, H(\mathsf{sk} \oplus \Delta_j, \rho_j) \oplus m_j)$ for fresh $\rho_j \leftarrow \{0,1\}^\lambda$; the ideal oracle answers with a fresh uniform string in $\{0,1\}^{2\lambda}$. Couple the two experiments by replacing each value $H(\mathsf{sk} \oplus \Delta_j, \rho_j)$ with a fresh and uniform u_j (then $(\rho_j, u_j \oplus m_j)$ is exactly a fresh uniform 2λ-bit string). The coupling fails only if some oracle point $(\mathsf{sk} \oplus \Delta_j, \rho_j)$ is evaluated twice across the experiment: (i) *two encryption queries collide*: this requires $\rho_j = \rho_{j'}$ (and $\Delta_j = \Delta_{j'}$) for some $j \neq j'$; since each ρ_j is fresh uniform, this has probability at most $q_E^2/2^{\lambda+1}$; (ii) *a direct query hits*: $\mathcal{A}$ queries H at some (x, ρ) with $x = \mathsf{sk} \oplus \Delta_j$ and $\rho = \rho_j$ for some j. In the ideal experiment the view of $\mathcal{A}$ is independent of sk, so each of the q_H direct queries hits one of the at most q_E points only if $\mathsf{sk} = x \oplus \Delta_j$ for one of the corresponding j's, an event of probability at most $q_E/2^\lambda$ over the uniform sk; a union bound gives $q_H q_E/2^\lambda$. An identical-until-bad argument completes the proof. □

LPN-based construction. For a noise parameter $0 < \epsilon < 1/2$, the decisional Learning Parity with Noise (LPN) assumption asserts that, for every polynomial $t = t(k)$, $(A, As \oplus e)$ is computationally indistinguishable from (A, u), where $A \leftarrow \mathbb{F}_2^{t \times k}$, the secret $s \leftarrow \mathbb{F}_2^k$, and $u \leftarrow \mathbb{F}_2^t$ are independently uniform, and $e \in \mathbb{F}_2^t$ is an independent noise vector whose bits are independently 1 with probability ϵ.

Theorem 2.9 (LPN instantiation, [AHI11])**.** *Assuming decisional LPN with noise parameter $\epsilon = 1/2 - \gamma$ where $1/\mathrm{poly}(k) \le \gamma \le 1/2$, there exists XOR-RKA-secure encryption with pseudorandom ciphertexts.*

Applebaum, Harnik, and Ishai [AHI11] state their result for any constant $0 < \gamma \le 1/2$, but we observe that the same construction works for $\gamma = 1/\mathrm{poly}$.

Indeed, the construction of [AHI11] encrypts m as $(A, As \oplus e \oplus C(m))$ for an error-correcting code C that allows efficient recovery from independent errors of rate ϵ. By taking the code to be a repetition code with λ/γ^2 repetitions, we get correctness with all but negligible probability.

2.3 Pseudorandom Functions

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

Definition 2.10 (Pseudorandom function)**.** *An efficiently computable family of functions $F = \left\{F_k \colon \{0,1\}^{n(\lambda)} \to \{0,1\}^{m(\lambda)}\right\}_{k \in \{0,1\}^\lambda}$ is a* pseudorandom function family (PRF) *if for every oracle-aided non-uniform PPT $\mathcal{A}$, there exists a negligible function μ, such that for any λ,*

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

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

3 Semi-Verifiable SK-$^1/_2$PIR

In this section we define and construct semi-verifiable linear-query SK-PIR.

Linear-Query SK-PIR. We start with defining our notion of (non-interactive) linear-query PIR, which is simply a PIR in which the database preprocessing is homomorphic with respect to linear operations.

Definition 3.1 (Secret-key linear-query PIR)**.** *A* secret-key linear-query private information retrieval (sk-LPIR) *consists of PPT algorithms* $(\mathsf{Gen}, \mathsf{Preprocess}, \mathsf{Query}, \mathsf{Answer}, \mathsf{Decode})$ *such that:*

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

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

- $\mathsf{Query}(\mathsf{sk}, M, i)$*: The query algorithm. Takes a secret key* sk*, a matrix* $M \in \{0,1\}^{N' \times N}$ *for some* $1 \leq N' \leq N$*, and an index* $i \in [N']$*, and outputs a client message* q *and a decoding key* $dec \in \{0,1\}^*$*.*

- $\mathsf{Answer}(\widehat{D}, M, q)$*: The server answer algorithm. Takes a database encoding* $\widehat{D}$*, a matrix* M *and a PIR-query* q *and returns an answer* a*.*

- $\mathsf{Decode}(a, i, dec)$*: The decoding algorithm. Takes a PIR-answer* a*, an index* i *and a decoding key* dec *and outputs a bit* $b \in \{0,1\}$*.*

Correctness: *The client outputs the correct value with overwhelming probability. For any* $N = \mathrm{poly}(\lambda)$ *there exists a negligible function* μ *such that the following holds. For every* $\lambda \in \mathbb{N}$*, database* $D \in \{0,1\}^N$*, full-rank matrix* $M \in \{0,1\}^{N' \times N}$ *with* $1 \leq N' \leq N$*, and index* $i \in [N']$*,*

$$\Pr[\mathsf{Decode}(a, i, dec) = (M \cdot D)[i]] \geq 1 - \mu(\lambda),$$

where the probability is over $\mathsf{sk} \leftarrow \mathsf{Gen}(1^\lambda, 1^N)$*,* $\widehat{D} \leftarrow \mathsf{Preprocess}(\mathsf{sk}, D)$*,* $(q, dec) \leftarrow \mathsf{Query}(\mathsf{sk}, M, i)$ *and* $a \leftarrow \mathsf{Answer}(\widehat{D}, M, q)$*.*

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

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

where $\mathsf{sk} \leftarrow \mathsf{Gen}(1^\lambda, 1^N)$ *and* $\widehat{D} \leftarrow \mathsf{Preprocess}(\mathsf{sk}, D)$ *are sampled once and fixed throughout the experiment, the queries of* $\mathcal{A}$ *are pairs* (M, i) *with* $M \in \{0,1\}^{N' \times N}$*,* $1 \leq N' \leq N$*, and* $i \in [N']$*,*

and each oracle answers every query with fresh independent randomness: $\mathsf{Query}_q(\mathsf{sk}, M, i)$ *samples* $(q, dec) \leftarrow \mathsf{Query}(\mathsf{sk}, M, i)$ *and outputs* q*, and* $\mathsf{Query}'_q(\mathsf{sk}, \cdot, \cdot)$ *is an oracle that given input* (M, i) *ignores* i *and outputs* $\mathsf{Query}_q(\mathsf{sk}, M, 1)$*.*

Semi-verifiable SK-1/2LPIR. We next define the notion of semi-verifiable SK-1/2LPIR, which we construct in the rest of this section.

Definition 3.2 (SK-1/2LPIR)**.** *An SK-1/2LPIR is defined as an SK-LPIR (with rectangular matrices* $M \in \{0,1\}^{N' \times N}$*,* $N' \leq N$*), except that* $\mathsf{Decode}(a, i, dec)$ *outputs a pair* $(b, \mathsf{status}) \in \{0,1\} \times \{\mathsf{success}, \mathsf{fail}\}$*.*

Correctness. *For any* $N = \mathrm{poly}(\lambda)$ *there exists a negligible function* μ *such that the following holds. For every* λ*,* $D \in \{0,1\}^N$*,* $N' \leq N$*, full-rank matrix* $M \in \{0,1\}^{N' \times N}$ *and* $i \in [N']$*:*

$$\Pr[b = (M \cdot D)[i]] \geq 1 - \mu(\lambda),$$

where $\mathsf{sk} \leftarrow \mathsf{Gen}(1^\lambda, 1^N)$*,* $\widehat{D} \leftarrow \mathsf{Preprocess}(\mathsf{sk}, D)$*,* $(q, dec) \leftarrow \mathsf{Query}(\mathsf{sk}, M, i)$*,* $a \leftarrow \mathsf{Answer}(\widehat{D}, M, q)$ *and* $(b, \mathsf{status}) \leftarrow \mathsf{Decode}(a, i, dec)$*.*

Balance. *For any* $N = \mathrm{poly}(\lambda)$ *there exists a negligible function* μ *such that the following holds. For every* λ*,* $D \in \{0,1\}^N$ *and a full-rank matrix* $M \in \{0,1\}^{N' \times N}$*, with probability at least* $1 - \mu(\lambda)$ *over* $\mathsf{sk} \leftarrow \mathsf{Gen}(1^\lambda, 1^N)$ *and* $\widehat{D} \leftarrow \mathsf{Preprocess}(\mathsf{sk}, D)$*, the pair* $(\mathsf{sk}, \widehat{D})$ *is* balanced for M*, meaning: for every* $i \in [N']$*,*

$$\left| \Pr_{q, dec, a, coins} \left[\mathsf{status} = \mathsf{success} \;\middle|\; \mathsf{sk}, \widehat{D} \right] - \tfrac{1}{2} \right| \leq \mu(\lambda),$$

where the conditional probability is over $(q, dec) \leftarrow \mathsf{Query}(\mathsf{sk}, M, i)$*,* $a \leftarrow \mathsf{Answer}(\widehat{D}, M, q)$*,* $(b, \mathsf{status}) \leftarrow \mathsf{Decode}(a, i, dec)$ *only.*

Semi-verifiable security. *An SK-1/2LPIR is* semi-verifiable *if there exist PPT algorithms* $Q_{\mathsf{success}}(\mathsf{sk}, \widehat{D}, M)$ *and* $Q_{\mathsf{fail}}(\mathsf{sk}, \widehat{D}, M, i)$ *(where* Q_{success} *does* not *receive the index) such that for every oracle-aided non-uniform PPT* $\mathcal{A}$ *and polynomial* $N = N(\lambda)$ *there is a negligible* μ *such that for every* λ *and* $D \in \{0,1\}^N$*,*

$$\left| \Pr\left[\mathcal{A}^{\mathcal{O}_{\mathsf{Client}}}(1^\lambda, \widehat{D}) = 1 \right] - \Pr\left[\mathcal{A}^{\mathcal{O}_{\mathsf{Sim}}}(1^\lambda, \widehat{D}) = 1 \right] \right| \leq \mu(\lambda),$$

where $\mathsf{sk} \leftarrow \mathsf{Gen}(1^\lambda, 1^N)$*,* $\widehat{D} \leftarrow \mathsf{Preprocess}(\mathsf{sk}, D)$*, and* $\mathcal{O}_{\mathsf{Client}}$*,* $\mathcal{O}_{\mathsf{Sim}}$ *are the interactive oracles (Definition 2.1) induced by the following session programs on session input* (M, i) *with* $M \in \{0,1\}^{N' \times N}$*,* $i \in [N']$*:*

- $\mathsf{Client}[\mathsf{sk}, \widehat{D}](M, i)$*: sample* $(q, dec) \leftarrow \mathsf{Query}(\mathsf{sk}, M, i)$*, send* q*; upon receiving* a*, compute* $(b, \mathsf{status}) \leftarrow \mathsf{Decode}(a, i, dec)$ *and send* status*. (The bit* b *is a local output, not sent.)*
- $\mathsf{Sim}[\mathsf{sk}, \widehat{D}](M, i)$*: sample* $\mathsf{status} \leftarrow \{\mathsf{success}, \mathsf{fail}\}$ *uniformly. If* $\mathsf{status} = \mathsf{success}$*: send* $q \leftarrow Q_{\mathsf{success}}(\mathsf{sk}, \widehat{D}, M)$*; upon receiving* a*, send* $\mathsf{success}$ *(ignoring* a*). If* $\mathsf{status} = \mathsf{fail}$*: sample* $(q, dec) \leftarrow Q_{\mathsf{fail}}(\mathsf{sk}, \widehat{D}, M, i)$*, send* q*; upon receiving* a*, compute* $(b, \mathsf{status}') \leftarrow \mathsf{Decode}(a, i, dec)$ *and send* status'*.*

3.1 Semi-Verifiable SK-1/2LPIR from XOR-RKA-Secure Encryption

We start by constructing a SK-1/2LPIR. We prove the following lemma.

Lemma 3.3. *Assume the existence of an XOR-RKA-secure encryption scheme with pseudorandom ciphertexts. Then there exists a semi-verifiable balanced SK-1/2LPIR with* $\mathrm{poly}(\lambda)$ *communication.*

Given a vector of keys $k = (k_1, \ldots, k_N) \in (\{0,1\}^\lambda)^N$ and a matrix $M \in \{0,1\}^{N' \times N}$, we define $M(k)_i = \sum_{j=1}^{N} M_{i,j} \cdot k_j$. Let $(\mathsf{Enc}, \mathsf{Dec})$ be an XOR-RKA secure encryption scheme with pseudorandom ciphertexts, and let $F\colon \{0,\ldots,2^\lambda\} \to \{0,1\}^\lambda$ be a PRF. We next describe our protocol. In the following, $+$ denotes bit-wise XOR.

The 1/2LPIR Protocol:

- **Key Generation:** $\mathsf{Gen}(1^\lambda, 1^N)$ outputs a random key $\mathsf{sk} \leftarrow \{0,1\}^\lambda$.
- **Preprocessing:** $\mathsf{Preprocess}(\mathsf{sk}, D)$ given a secret key sk and a database $D \in \{0,1\}^N$:
  1. For any $i \in [N]$, computes $k_{i,0} = F_{\mathsf{sk}}(i)$ and $k_{i,1} = F_{\mathsf{sk}}(i) \oplus \Delta$, where $\Delta = F_{\mathsf{sk}}(0)$.
  2. Outputs $\widehat{D} = \{(k_{i,b}, b) : i \in [N], b = D[i]\}$
- **Query:** $\mathsf{Query}(\mathsf{sk}, M, i^*)$ given a secret key sk, a matrix $M \in \{0,1\}^{N' \times N}$ and index i^*:
  1. Reconstructs $k_{j,0} = F_{\mathsf{sk}}(j)$ and $\Delta = F_{\mathsf{sk}}(0)$ from sk.
  2. For every $i \in [N']$, let key $k'_{i,0} = M(k)_i$ for $k = (k_{1,0}, \ldots, k_{N,0})$, and $k'_{i,1} = k'_{i,0} + \Delta$.
  3. Samples $b \leftarrow \{0,1\}, r \leftarrow \{0,1\}^\lambda$ and computes $c = \mathsf{Enc}(k'_{i^*,b}, r)$.
  4. Outputs a client message $q = (r, c)$ and a decoding key $dec = (b, k'_{i^*,b})$.
- **Answer:** $\mathsf{Answer}(\widehat{D}, M, q)$ given a database encoding $\widehat{D}$, a matrix M and a query $q = (r, c)$:
  1. For every $i \in [N']$ let $k'_i = M(k)_i$ for $k = (k_{1,D[1]}, \ldots, k_{N,D[N]})$.
  2. If for some $i \in [N']$ it holds that $\mathsf{Dec}(k'_i, c) = r$, Answer outputs k'_i for the smallest such i. Otherwise Answer outputs $\perp$.
- **Decoding:** $\mathsf{Decode}(a, i, dec)$ given an answer $a \in \{0,1\}^*$, an index i and a decoding key $dec = (b, k)$, outputs (b, fail) if $k = a$ or $(1-b, \mathsf{success})$ otherwise.

Note that the Query algorithm can compute only the key $k'_{i^*,0} = \bigoplus_{j \in \mathrm{supp}(M_{i^*})} F_{\mathsf{sk}}(j)$ of the queried row, in time $|\mathrm{supp}(M_{i^*})| \cdot \mathrm{poly}(\lambda)$.

Correctness. We start with proving the correctness of the protocol. We will need the following claim.

Claim 3.4. *Assume* $(\mathsf{Enc}, \mathsf{Dec})$ *is XOR-RKA-secure with pseudorandom ciphertexts. Then for any* $N = \mathrm{poly}(\lambda)$ *there exists a negligible function* μ *such that the following holds. For every* λ*,* $D \in \{0,1\}^N$*, full-rank matrix* $M \in \{0,1\}^{N' \times N}$*,* $i, i^* \in [N']$ *and* $b, b^* \in \{0,1\}$ *with* $(i,b) \neq (i^*, b^*)$*,*

$$\Pr\left[\mathsf{Dec}\left(k'_i \oplus b \cdot \Delta, \mathsf{Enc}(k'_{i^*} \oplus b^* \cdot \Delta,\, r)\right) = r\right] \leq \mathrm{neg}(\lambda)$$

where $\mathsf{sk} \leftarrow \mathsf{Gen}(1^\lambda, 1^N)$*,* $\widehat{D} \leftarrow \mathsf{Preprocess}(\mathsf{sk}, D)$*,* $r \leftarrow \{0,1\}^\lambda$*, the keys* $k'_j = M(k)_j$ *(for* $k = (k_{1,D[1]}, \ldots, k_{N,D[N]})$*) are as computed by* Answer*, and* $\Delta = F_{\mathsf{sk}}(0)$*.*

Proof. Consider the hybrid experiment H in which the $N+1$ values $k_{1,0},\ldots,k_{N,0},\Delta$ are replaced by uniform independent strings in $\{0,1\}^\lambda$. The event we claim to happen with negligible probability is efficiently testable given oracle access to either F_{sk} or a uniformly random function: the tester queries its oracle on $0,1,\ldots,N$, computes all derived keys, samples r and the encryption coins, and checks the decryption equation. Hence, by PRF security (against non-uniform adversaries, with D, M, i, i^* hardwired), the probability of each event changes by at most $\mathrm{neg}(\lambda)$ between the real experiment and H.

Consider now the hybrid H. Let $x_j := k_{j,D[j]} = k_{j,0} \oplus D[j]\cdot\Delta$. The map $(k_{1,0},\ldots,k_{N,0},\Delta) \mapsto (x_1,\ldots,x_N,\Delta)$ is a bijection of $(\{0,1\}^\lambda)^{N+1}$, so $(x_1,\ldots,x_N,\Delta)$ is uniform. Note $k'_j = \bigoplus_u M_{j,u}x_u$ is a deterministic function of x. Moreover, since M is a full-rank matrix,$(k'_1,\ldots,k'_{N'},\Delta)$ is uniform. Therefore, for any choice of $(i,b)\neq(i^*,b^*)$, the pair $(k'_i \oplus b\cdot\Delta, k'_{i^*}\oplus b^*\cdot\Delta)$ is uniformly distributed.

It is thus enough to prove that for any pair of randomly distributed keys (k_1,k_2) it holds that

$$p := \Pr\big[\mathsf{Dec}\big(k_1,\mathsf{Enc}(k_2,\,r)\big)=r\big] \le \mathrm{neg}(\lambda),$$

which holds by the security of the encryption scheme. Indeed, assume this is not the case; we construct an algorithm that breaks the pseudorandomness of the encryption. Consider the algorithm $\mathcal{A}$ that, given oracle access to $\mathsf{Enc}(RK(k_1,\cdot),\cdot)$, chooses $r\leftarrow\{0,1\}^\lambda$ and a secret key $k_2\leftarrow\{0,1\}^\lambda$, and obtains $e=\mathsf{Enc}(k_1,r)$ by querying the oracle on input $(0^\lambda,r)$. It then outputs 1 if $\mathsf{Dec}(k_2,e)=r$.

When $k_1\leftarrow\{0,1\}^\lambda$, the probability that $\mathcal{A}$ outputs 1 is exactly p. Next, let Π be the oracle that returns a random answer for each query. It is clear that $\mathcal{A}^\Pi(1^\lambda)$ outputs 1 with probability $2^{-\lambda}$. Thus, by the security of the encryption, $\mathcal{A}^{\mathsf{Enc}(RK(k_1,\cdot),\cdot)}(1^\lambda)$ must output 1 with negligible probability. □

We are now ready to prove the correctness of the protocol.

Claim 3.5. *The protocol is correct and balanced.*

Proof. First, notice that for every $i\in[N']$, the value of k'_i computed in Answer is equal to the value of $k'_{i,c}$ computed in Query, for $c=(M\cdot D)_i$. Indeed,

$$k'_i = \sum_{j=1}^{N} M_{i,j}\cdot k_{j,D[j]} = \sum_{j=1}^{N} M_{i,j}\cdot(k_{j,0}+D[j]\cdot\Delta) = k'_{i,0} + \Delta\cdot(M\cdot D)_i,$$

which equals to $k'_{i,0}$ if $(M\cdot D)_i=0$ or to $k'_{i,1}$ otherwise.

Fix D, M, i^*, and let Err denote the event that either (1) $b=(M\cdot D)_{i^*}$ and Answer does not return k'_{i^*}, or (2) $b\neq(M\cdot D)_{i^*}$ and Answer does not return $\perp$. By Claim 3.4 and correctness of $(\mathsf{Enc},\mathsf{Dec})$, $\Pr[\mathsf{Err}]\le\mu_0(\lambda)$ for a negligible μ_0. Conditioned on $\neg\mathsf{Err}$, Decode always outputs the correct value, and moreover, $\mathsf{status}=\mathsf{success}$ iff $b\neq(M\cdot D)_{i^*}$. Since b is random and independent of $(\mathsf{sk},\widehat{D})$, we have for every fixed $(\mathsf{sk},\widehat{D})$:

$$\left|\Pr\left[\mathsf{status}=\mathsf{success}\mid\mathsf{sk},\widehat{D}\right]-\tfrac{1}{2}\right| \le \Pr\left[\mathsf{Err}\mid\mathsf{sk},\widehat{D}\right].$$

Let $p_{i^*}(\mathsf{sk},\widehat{D}) := \Pr\left[\mathsf{Err}\mid\mathsf{sk},\widehat{D}\right]$; then $\mathrm{E}[p_{i^*}]\le\mu_0(\lambda)$, and by a union bound over the at most $N'\le N$ indices i^*, $\mathrm{E}[\sum_{i^*}p_{i^*}]\le N\mu_0$. By Markov,

$$\Pr_{\mathsf{sk},\widehat{D}}\left[\exists i^*\colon p_{i^*}(\mathsf{sk},\widehat{D})>\sqrt{N\mu_0(\lambda)}\right] \le \Pr_{\mathsf{sk},\widehat{D}}\left[\sum_{i^*}p_{i^*}(\mathsf{sk},\widehat{D})>\sqrt{N\mu_0(\lambda)}\right] \le \sqrt{N\mu_0(\lambda)} = \mathrm{neg}(\lambda).$$

This concludes the proof.

□

Security. We next prove the security of the protocol. We describe the simulator.

The simulation oracles Q_{fail} and Q_{success}. We next define the oracles Q_{fail} and Q_{success}. The oracle $Q_{\mathsf{success}}(\mathsf{sk}, \widehat{D}, M)$ samples and outputs a uniformly random value $q \leftarrow \{0,1\}^{\lambda+\ell(\lambda)}$ (where $\ell(\lambda)$ is the output length of $\mathsf{Enc}\colon \{0,1\}^\lambda \times \{0,1\}^\lambda \to \{0,1\}^{\ell(\lambda)}$). The oracle $Q_{\mathsf{fail}}(\mathsf{sk}, \widehat{D}, M, i)$ behaves like the real query algorithm, with its internal bit fixed to $b = (M \cdot D)_i$: it samples the remaining randomness exactly as Query and outputs the resulting (q, dec). Here D can be recovered from $\widehat{D}$, which contains the pairs $(k_{j,D[j]}, D[j])$.

Claim 3.6. *The protocol is semi-verifiable (with the algorithms $Q_{\mathsf{success}}, Q_{\mathsf{fail}}$ from above).*

Proof. We prove the claim by a hybrid argument. Fix an oracle-aided efficient adversary $\mathcal{A}$.

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

Hybrid 1: Random Symmetric Encryption Keys. In this hybrid, all keys $k_{i,0}$, as well as Δ, are chosen truly at random instead of being derived using the PRF F_{sk}. Since $\mathcal{A}$ does not obtain sk, this hybrid is indistinguishable from the previous one by the security of the PRF.

Let $x_j := k_{j,D[j]} = k_{j,0} \oplus D[j] \cdot \Delta$. Since the map $(k_{1,0}, \ldots, k_{N,0}, \Delta) \mapsto (x_1, \ldots, x_N, \Delta)$ is a bijection of $(\{0,1\}^\lambda)^{N+1}$, the distribution of $(x_1, \ldots, x_N, \Delta)$ in this hybrid is uniform.

Hybrid 2: Always answer $\mathsf{success}$ when $b \neq (M \cdot D)_i$. In this hybrid, whenever the bit b chosen by Query is $1 - (M \cdot D)_i$, the status output by the oracle is $\mathsf{success}$, independently of the message a sent by $\mathcal{A}$. The two hybrids differ only if, in a session with $b \neq (M \cdot D)_i$, $\mathcal{A}$ returns the decoding key $k'_{i,b} = k'_i + \Delta$. Below, we claim that finding this key with non-negligible probability is equivalent to guessing Δ with non-negligible probability, which contradicts the XOR-RKA security.

Indeed, let Bad be the event that in some session with $b \neq (M \cdot D)_i$, the adversary's answer a equals the session's decoding key $k'_i \oplus \Delta$. Hybrids 1 and 2 are identical until Bad, hence their distinguishing advantage is at most $\Pr[\mathsf{Bad}]$ (in Hybrid 2). Suppose $\Pr[\mathsf{Bad}] = \delta(\lambda)$ and let $s = s(\lambda)$ bound the number of sessions opened by $\mathcal{A}$. We build an RKA adversary $\mathcal{B}$ with advantage at least $\delta/s - 2^{-\lambda} - \mathrm{neg}$:

- $\mathcal{B}$ has access to an oracle $O(\cdot, \cdot)$ which is either $\mathsf{Enc}(\Delta \oplus \phi, \cdot)$ for a hidden uniform Δ (real) or uniform strings (ideal). $\mathcal{B}$ samples $x_1, \ldots, x_N \leftarrow \{0,1\}^\lambda$ itself, sets $\widehat{D} = ((x_j, D[j]))_{j \in [N]}$, and runs $\mathcal{A}(1^\lambda, \widehat{D})$, simulating Hybrid 2 as follows. For a session with input (M, i): it computes $k'_i = M(x)_i$; with probability $1/2$ it plays the fail branch (sample r, send $(r, \mathsf{Enc}(k'_i, r))$ computed locally, decode honestly against the known key k'_i); with probability $1/2$ it plays the ok branch: it samples r, queries $c \leftarrow O(\phi_{k'_i}, r)$, sends (r, c), and always reports status $\mathsf{success}$. In the real game this is exactly Hybrid 2.
- $\mathcal{B}$ picks a uniformly random ok-branch session j^* in advance; when $\mathcal{A}$ answers that session with some a, $\mathcal{B}$ sets $\widehat{\Delta} := a \oplus k'_i$ (for that session's derived key). It then samples a fresh $m^* \leftarrow \{0,1\}^\lambda$, queries $c^* \leftarrow O(\phi_{0^\lambda}, m^*)$, and outputs 1 iff $\mathsf{Dec}(\widehat{\Delta}, c^*) = m^*$.

- In the real game, with probability at least δ/s the guessed session realizes Bad, so $\widehat{\Delta} = \Delta$ and the test passes with probability $1 - \mathrm{neg}$ by decryption correctness (for the uniform key Δ). In the ideal game, c^* is uniform and independent of m^*, so the test passes with probability at most $2^{-\lambda}$ — note this uses that m^* is fresh and never otherwise revealed. Hence $\mathcal{B}$'s advantage is at least $\delta/s - \mathrm{neg} - 2^{-\lambda}$, so $\delta \leq s \cdot (\mathsf{Adv}_{\mathcal{B}} + 2^{-\lambda} + \mathrm{neg}) = \mathrm{neg}(\lambda)$.

Hybrid 3: Random Answers when $b \neq (M \cdot D)_i$. In this hybrid, whenever the bit b chosen by Query is $1 - (M \cdot D)_i$, q is sampled from the uniform distribution U over $\lambda + |\mathsf{Enc}(k, r)|$ bits. Indistinguishability follows by the same simulation as for the previous hybrid, except no extraction: $\mathcal{B}'$ simulates as above (fail branches locally, ok branches via $O(\phi_{k'_i}, r)$) and outputs whatever $\mathcal{A}$ outputs. If O is the real RKA oracle, the simulated experiment is Hybrid 2; if O is the ideal (uniform) oracle, each ok-branch query is (r, c) with r, c uniform, which is Hybrid 3.

Hybrid 4: Back to Pseudorandom Symmetric Encryption Keys. In this hybrid, all keys $k_{i,b}$, as well as Δ derived using the PRF F_{sk}. Since $\mathcal{A}$ does not obtain sk, this hybrid is indistinguishable from the previous one by the security of the PRF. Note that this hybrid is identical to $\mathcal{A}$ having oracle to Sim.

□

Proving Lemma 3.3.

Proof of Lemma 3.3. First, the assumed encryption scheme implies the existence of one-way functions and thus of PRFs. The correctness and balance of the scheme hold by Claim 3.5 and the security by Claim 3.6. The communication complexity holds by construction. □

4 Batch Random-Index LPIR

In this section we define and construct batch random-index LPIR.

Definition 4.1 (ϵ-secure t-batch Random-Index SK-LPIR)**.** *Let $t = t(\lambda, N)$ and $\epsilon = \epsilon(\lambda, N) \in [0,1]$. A t-batch random-index secret-key linear-query PIR (SK-RI-LPIR) consists of PPT algorithms $(\mathsf{Gen}, \mathsf{Preprocess}, \mathsf{Query}, \mathsf{Answer}, \mathsf{Decode})$ with the syntax above, amended as follows: $\mathsf{Query}(\mathsf{sk}, M)$ takes $M \in \{0,1\}^{N' \times N}$ ($1 \le N' \le N$); $\mathsf{Decode}(a, dec)$ outputs t pairs $\big((i_1, b_1), \ldots, (i_t, b_t)\big) \in ([N'] \times \{0,1\})^t$.*

Correctness. *For any $N = \mathrm{poly}(\lambda)$ there exists a negligible function μ such that the following holds. For every λ, $D \in \{0,1\}^N$ and full-rank matrix $M \in \{0,1\}^{N' \times N}$, in the honest experiment ($\mathsf{sk} \leftarrow \mathsf{Gen}$, $\widehat{D} \leftarrow \mathsf{Preprocess}(\mathsf{sk}, D)$, $(q, dec) \leftarrow \mathsf{Query}(\mathsf{sk}, M)$, $a \leftarrow \mathsf{Answer}(\widehat{D}, M, q)$, $\big((i_j, b_j)\big)_j \leftarrow \mathsf{Decode}(a, dec)$) the output is correct:*

$$\Pr[\forall j \in [t] \colon b_j = (M \cdot D)[i_j]] \ge 1 - \mu(\lambda).$$

Index uniformity. *For any $N = \mathrm{poly}(\lambda)$ there exists a negligible function μ such that the following holds for every λ, $D \in \{0,1\}^N$ and a full-rank matrix $M \in \{0,1\}^{N' \times N}$, with probability at least $1 - \mu(\lambda)$ over $\mathsf{sk} \leftarrow \mathsf{Gen}(1^\lambda, 1^N)$ and $\widehat{D} \leftarrow \mathsf{Preprocess}(\mathsf{sk}, D)$. In the honest experiment ($(q, dec) \leftarrow \mathsf{Query}(\mathsf{sk}, M)$, $a \leftarrow \mathsf{Answer}(\widehat{D}, M, q)$, $\big((i_j, b_j)\big)_j \leftarrow \mathsf{Decode}(a, dec)$):*

$$\mathrm{SD}\big((i_1, \ldots, i_t),\, U_{[N']^t}\big) \le \mu(\lambda),$$

where $U_{[N']^t}$ is uniform on $[N']^t$.

ϵ-security. *There exist interactive PPT session programs Sim' and Sim, where $\mathsf{Sim}'[\mathsf{sk}, \widehat{D}]$ on session input M interacts with the server and, at the point where the honest client would send its final message, produces a local value $\mathsf{flag} \in \{\bot\} \cup [N']^t$; and Sim is identical to Sim' except that as its final session message it sends flag if $\mathsf{flag} \neq \bot$, and otherwise a fresh uniform tuple $(i'_1, \ldots, i'_t) \leftarrow [N']^t$ sampled independently of everything else. We require:*

1. ($\bot$-frequency, statistical, aux-input) *For every λ, $N = \mathrm{poly}(\lambda)$, every sk in the support of $\mathsf{Gen}(1^\lambda, 1^N)$, every $D \in \{0,1\}^N$, every $\widehat{D}$ in the support of $\mathsf{Preprocess}(\mathsf{sk}, D)$, every $M \in \{0,1\}^{N' \times N}$, and every computationally* unbounded *interactive server strategy Server^* (which may depend on $\mathsf{sk}, D, \widehat{D}$): in a single session of $\mathsf{Sim}'[\mathsf{sk}, \widehat{D}]$ on input M against Server^*,*

   $$\Pr[\mathsf{flag} \neq \bot] \le \epsilon,$$

   the probability being over the session's coins only.

2. (Indistinguishability) *For every oracle-aided non-uniform PPT $\mathcal{A}$ and polynomial $N = N(\lambda)$ there is a negligible μ such that for every λ, $D \in \{0,1\}^N$:*

   $$\Big|\Pr\Big[\mathcal{A}^{\mathcal{O}_{\mathsf{Client}}}(1^\lambda, \widehat{D}) = 1\Big] - \Pr\Big[\mathcal{A}^{\mathcal{O}_{\mathsf{Sim}}}(1^\lambda, \widehat{D}) = 1\Big]\Big| \le \mu(\lambda),$$

   where $\mathcal{O}_{\mathsf{Client}}$, $\mathcal{O}_{\mathsf{Sim}}$ are the interactive oracles induced by the session programs $\mathsf{Client}[\mathsf{sk}, \widehat{D}](M)$ (as in the current definition: send $q \leftarrow \mathsf{Query}$, receive a, decode, send $(i_1, \ldots, i_t)$) and $\mathsf{Sim}[\mathsf{sk}, \widehat{D}](M)$, with $\mathsf{sk} \leftarrow \mathsf{Gen}(1^\lambda, 1^N)$, $\widehat{D} \leftarrow \mathsf{Preprocess}(\mathsf{sk}, D)$.

4.1 Semi-verifiable 1/2LPIR to Batch SK-RI-LPIR

We prove the following lemma.

Lemma 4.2. *Assume the existence of balanced semi-verifiable SK-1/2LPIR with communication c. Then, for every efficiently computable $t, 1/\epsilon \in \text{poly}(N, \lambda)$, there exists an ϵ-secure t-batch SK-RI-LPIR protocol with communication $O(c \cdot \text{polylog}(\lambda) \cdot t^2/\epsilon^2)$.*

We next describe our protocol. Fix t and ϵ, and let $(\mathsf{SV.Gen}, \mathsf{SV.Preprocess}, \mathsf{SV.Query}, \mathsf{SV.Answer}, \mathsf{SV.Decode})$ be a semi-verifiable 1/2LPIR.

The t-Batch SK-RI-LPIR Protocol:

- **Key Generation:** $\mathsf{Gen}(1^\lambda, 1^N)$ outputs a random key $\mathsf{sk} \leftarrow \mathsf{SV.Gen}(1^\lambda, 1^N)$.
- **Preprocessing:** $\mathsf{Preprocess}(\mathsf{sk}, D)$ given a secret key sk and a database $D \in \{0,1\}^N$, outputs $\widehat{D} \leftarrow \mathsf{SV.Preprocess}(\mathsf{sk}, D)$.
- **Query:** $\mathsf{Query}(\mathsf{sk}, M)$ given a secret key sk, and a matrix $M \in \{0,1\}^{N' \times N}$:
  1. Samples $\ell = 256t^2/\epsilon^2 \cdot \log^2 \lambda$ random indices $i_1, \ldots, i_\ell \leftarrow [N']$.
  2. Computes $(q_j, dec_j) \leftarrow \mathsf{SV.Query}(\mathsf{sk}, M, i_j)$ for every $j \in [\ell]$.
  3. Outputs $q = (q_1, \ldots, q_\ell)$, $dec = (i_1, \ldots, i_\ell, dec_1, \ldots, dec_\ell)$.
- **Answer:** $\mathsf{Answer}(\widehat{D}, M, q)$ given a database encoding $\widehat{D}$, a matrix M and a query $q = (q_1, \ldots, q_\ell)$:
  1. For every $j \in [\ell]$ let $a_j \leftarrow \mathsf{SV.Answer}(\widehat{D}, M, q_j)$.
  2. Answer outputs $a = (a_1, \ldots, a_\ell)$.
- **Decoding:** $\mathsf{Decode}(a, dec)$ given an answer $a = (a_1, \ldots, a_\ell)$, and a decoding key $dec = (i_1, \ldots, i_\ell, dec_1, \ldots, dec_\ell)$:
  1. Computes for each $j \in [\ell]$ $(b_j, \mathsf{status}_j) \leftarrow \mathsf{SV.Decode}(a_j, i_j, dec_j)$.
  2. Let $\mathcal{G} = \{j \in [\ell] : \mathsf{status}_j = \mathsf{success}\}$.
  3. If $||\mathcal{G}| - \ell/2| > \sqrt{\ell} \log \lambda$, Decode outputs $((i'_1, b'_1), \ldots, (i'_t, b'_t))$ for random $i'_1, \ldots, i'_t \leftarrow [N']$ and $b'_1, \ldots, b'_t \leftarrow \{0,1\}$.
  4. Otherwise, Decode samples t distinct elements $j_1, \ldots, j_t \leftarrow \mathcal{G}$ and outputs $((i_{j_1}, b_{j_1}), \ldots, (i_{j_t}, b_{j_t}))$.

Observe that, when using the Semi-verifiable 1/2LPIR protocol from Section 3, the running time of the client is $\ell \cdot k \cdot \text{poly}(\lambda)$, where $k = \max_i |\text{supp}(M_i)|$, and the server running time is $(\ell \cdot N' + |\text{supp}(M)|)\text{poly}(\lambda)$.

Remark 4.3 (Reducing server work with XOR-RKA PRFs)**.** *The $\ell N'$ term above, which will dominate the server's work in our final constant-round protocol (e.g., when $N' = N$ and M is the identity matrix), comes from testing each of the ℓ ciphertexts of a round against every one of the N' current keys. This can be avoided by replacing the encryption scheme* $(\mathsf{Enc}, \mathsf{Dec})$ *with an XOR-RKA-secure PRF F (available in the random-oracle model), as in the technical overview: a 1/2LPIR query for key k becomes a pair (x, y) with $y = F(k, x)$, and the decoder declares a match*

whenever $y = F(k', x)$ for one of the server's keys k'. If, in addition, the ℓ queries share one nonce *x, the server can compute $F(k'_i, x)$ once for each of its N' keys, store the results in a table, and answer each of the ℓ queries with a single lookup, at a total cost of $N' + \ell$ instead of $\ell N'$ per round.*

The analysis can be extended to this variant with minor adaptations: first, the ℓ indices sampled for a round's batch must be distinct rather than i.i.d. uniform (with a shared x, two queries at the same index would produce identical tags y, which cannot be simulated). Second, the semi-verifiable security of Lemma 3.3 and the ϵ-security of the batch random-index LPIR (Definition 4.1) need to be stated for the whole round (one session with ℓ sub-queries sharing x) rather than session by session; the hybrid argument of Claim 3.6 then extends directly, since XOR-RKA security already allows the adversary to query its oracle at the same input under many different key offsets. We note that this optimization does not carry to the LPN settings, as XOR-RKA PRFs are not known from LPN.

Correctness. We start with the correctness of the protocol. We prove the following claim.

Claim 4.4. *The protocol is correct.*

Proof. Fix a database D and a full-rank matrix $M \in \{0,1\}^{N' \times N}$, and consider an honest execution of the protocol with random $\mathsf{sk} \leftarrow \mathsf{Gen}(1^\lambda, 1^N)$ and $\widehat{D} \leftarrow \mathsf{Preprocess}(\mathsf{sk}, D)$. Let $(I_1, \ldots, I_\ell)$, $(B_1, \ldots, B_\ell)$ and $(S_1, \ldots, S_\ell)$ be the random variables taking the values of $(i_1, \ldots, i_\ell)$, $(b_1, \ldots, b_\ell)$, and $(\mathsf{status}_1, \ldots, \mathsf{status}_\ell)$, resp., as retrieved by Decode in this execution. Observe that the output of Decode is determined by these values and its internal randomness.

In the following, we move to a hybrid H in which $B_j = MD[I_j]$ for any j, and $S_j \leftarrow \{\mathsf{success}, \mathsf{fail}\}$ is chosen uniformly and independently for any j (and independently of the secret key). By the negligible correctness and balance error of SV, the output of Decode in this hybrid is statistically close to its output in the real experiment. Consequently, it suffices to analyze the correctness of the protocol in H.

Let $\mathcal{G} = \{j : S_j = \mathsf{success}\}$. By Chernoff,

$$\Pr\left[\left||\mathcal{G}| - \ell/2\right| > \sqrt{\ell}\log\lambda\right] = \mathrm{neg}(\lambda).$$

If this event does not occur, the decoder samples t positions from $\mathcal{G}$ and every returned bit is equal to the corresponding entry of $M \cdot D$. Observe that when this event does not occur, the choice of ℓ ensures $|\mathcal{G}| \geq t$. If the event occurs, the decoder may output random bits, but this happens only with negligible probability. This proves output correctness.

It remains to prove index uniformity. Condition on any fixed set $\mathcal{G}$ with $\left||\mathcal{G}| - \ell/2\right| \leq \sqrt{\ell}\log\lambda$, and on any distinct positions $j_1, \ldots, j_t \in \mathcal{G}$ selected by the decoder. Since in H the variables $I_1, \ldots, I_\ell$ are independent uniform elements of $[N']$ and are independent of $\mathcal{G}$, the tuple

$$(I_{j_1}, \ldots, I_{j_t})$$

is distributed exactly as $U_{[N']^t}$. In the other branch, the decoder explicitly samples a fresh uniform tuple from $[N']^t$. Thus, in hybrid H, the output-index tuple is exactly uniform over $[N']^t$. Since the real experiment and H are statistically close, the output indices in the real experiment are negligibly close to $U_{[N']^t}$, as required.

□

Security. We move to analyze the security of the protocol. Consider the following simulator.

The Simulator: $\mathsf{Sim}'[\mathsf{sk}, \widehat{D}](M)$. Let $Q_{\mathsf{success}}, Q_{\mathsf{fail}}$ be the algorithms promised by the security of the 1/2PIR. Given a matrix $M \in \{0,1\}^{N' \times N}$, $\mathsf{Sim}'[\mathsf{sk}, \widehat{D}]$ acts as follows.

- **Query:**
  1. For $\ell = 256t^2/\epsilon^2 \cdot \log^2 \lambda$, Sim' samples $\mathsf{status}_j \leftarrow \{\mathsf{success}, \mathsf{fail}\}$ for each $j \in [\ell]$.
  2. For each $j \in [\ell]$ with $\mathsf{status}_j = \mathsf{fail}$, Sim' samples a random index $i_j \leftarrow [N']$.
  3. For each $j \in [\ell]$ with $\mathsf{status}_j = \mathsf{success}$, let $q_j \leftarrow Q_{\mathsf{success}}(\mathsf{sk}, \widehat{D}, M)$. For each $j \in [\ell]$ with $\mathsf{status}_j = \mathsf{fail}$, let $(q_j, dec_j) \leftarrow Q_{\mathsf{fail}}(\mathsf{sk}, \widehat{D}, M, i_j)$.
  4. Outputs $q = (q_1, \ldots, q_\ell)$.
- **Decoding:** Given an answer $a = (a_1, \ldots, a_\ell)$:
  1. Sim' computes for each $j \in [\ell]$ with $\mathsf{status}_j = \mathsf{fail}$ $(b_j, \mathsf{status}'_j) \leftarrow \mathsf{SV.Decode}(a_j, i_j, dec_j)$. If $\mathsf{status}_j = \mathsf{success}$, it sets $\mathsf{status}'_j = \mathsf{success}$.
  2. Let $\mathcal{G} = \left\{ j \in [\ell] \colon \mathsf{status}'_j = \mathsf{success} \right\}$.
  3. If $||\mathcal{G}| - \ell/2| > \sqrt{\ell} \log \lambda$, Sim' outputs $\mathsf{flag} = \perp$.
  4. Otherwise, Sim' samples t distinct elements $j_1, \ldots, j_t \leftarrow \mathcal{G}$.
  5. If for every $k \in [t]$, $\mathsf{status}_{j_k} = \mathsf{success}$, Sim' outputs $\mathsf{flag} = \perp$.
  6. Otherwise, for each $j \in [\ell]$ with $\mathsf{status}_j = \mathsf{success}$, Sim' samples a random index $i_j \leftarrow [N']$. Sim' outputs $\mathsf{flag} = (i_{j_1}, \ldots, i_{j_t})$.

Claim 4.5. *For every oracle-aided non-uniform PPT $\mathcal{A}$ and polynomial $N = N(\lambda)$ there is a negligible μ such that for every λ, $D \in \{0,1\}^N$:*

$$\left| \Pr\left[\mathcal{A}^{\mathcal{O}_{\mathsf{Client}}}(1^\lambda, \widehat{D}) = 1 \right] - \Pr\left[\mathcal{A}^{\mathcal{O}_{\mathsf{Sim}}}(1^\lambda, \widehat{D}) = 1 \right] \right| \leq \mu(\lambda),$$

where $\mathcal{O}_{\mathsf{Client}}$, $\mathcal{O}_{\mathsf{Sim}}$ are as defined in Definition 4.1, for $\mathsf{sk} \leftarrow \mathsf{Gen}(1^\lambda, 1^N)$, $\widehat{D} \leftarrow \mathsf{Preprocess}(\mathsf{sk}, D)$.

Proof. Let $\mathsf{SV}.\mathcal{O}_{\mathsf{Client}}$ and $\mathsf{SV}.\mathcal{O}_{\mathsf{Sim}}$ be the two oracles from the semi-verifiable security of SV, where $\mathsf{SV}.\mathcal{O}_{\mathsf{Sim}}$ is defined with respect to the algorithms $Q_{\mathsf{success}}, Q_{\mathsf{fail}}$ used by Sim'. Consider the following oracle-aided interactive algorithm $\mathcal{B}$. On a session with input M, $\mathcal{B}$ samples $i_1, \ldots, i_\ell \leftarrow [N']$ and opens ℓ sessions of its oracle, on inputs $(M, i_1), \ldots, (M, i_\ell)$. It sends the first messages of these sessions as its query $q = (q_1, \ldots, q_\ell)$, and given an answer $a = (a_1, \ldots, a_\ell)$, it forwards a_j to the j-th session and gets back a status status_j. Finally, $\mathcal{B}$ computes its last message as Decode does: it lets $\mathcal{G} = \{j \colon \mathsf{status}_j = \mathsf{success}\}$, sends t random indices if $||\mathcal{G}| - \ell/2| > \sqrt{\ell} \log \lambda$, and otherwise samples t distinct $j_1, \ldots, j_t \leftarrow \mathcal{G}$ and sends $(i_{j_1}, \ldots, i_{j_t})$.

$\mathcal{B}^{\mathsf{SV}.\mathcal{O}_{\mathsf{Client}}}$ behaves exactly as $\mathcal{O}_{\mathsf{Client}}$ (the j-th session of the oracle is simply the j-th execution of SV inside Client) and $\mathcal{B}^{\mathsf{SV}.\mathcal{O}_{\mathsf{Sim}}}$ behaves exactly as $\mathcal{O}_{\mathsf{Sim}}$: the only difference is when the indices of the $\mathsf{success}$ sessions are sampled ($\mathcal{B}$ samples all of them in advance, while Sim' samples them only at the end, and only if one of the selected sessions is a fail session; else Sim sends fresh random indices). But Q_{success} does not get the index as input, so the index of a $\mathsf{success}$ session is not used anywhere before the last message, and it is uniform and independent of everything else. Hence, sampling it in advance, at the end, or replacing it by a fresh random index makes no difference.

The claim now follows from the semi-verifiable security of SV: $\mathcal{A}$ together with $\mathcal{B}$ is an efficient algorithm that distinguishes $\mathsf{SV}.\mathcal{O}_{\mathsf{Client}}$ from $\mathsf{SV}.\mathcal{O}_{\mathsf{Sim}}$ with the same advantage that $\mathcal{A}$ has in distinguishing $\mathcal{O}_{\mathsf{Client}}$ from $\mathcal{O}_{\mathsf{Sim}}$. □

Proving Lemma 4.2.

Proof of Lemma 4.2. The correctness of the scheme follows by Claim 4.4. It remains to verify the ϵ-security guarantee of the simulator. Indistinguishability follows from Claim 4.5. Let $T = \{j \in [\ell] : \mathsf{status}_j = \mathsf{success}\}$ be the set of true-success indices. Using Chernoff bound we get that $|T| \geq \ell/2 - \sqrt{\ell}\log\lambda$ except with negligible probability. Moreover, $T \subseteq \mathcal{G}$. If $||\mathcal{G}| - \ell/2| > \sqrt{\ell}\log\lambda$, then Sim' outputs $\perp$ immediately. Otherwise,

$$|\mathcal{G} \setminus T| \leq 2\sqrt{\ell}\log\lambda.$$

Since $|\mathcal{G}| \geq |T| \geq \ell/2 - \sqrt{\ell}\log\lambda$, a union bound over the t sampled indices shows that, for sufficiently large λ, the probability of sampling an index outside T is at most

$$t \cdot \frac{2\sqrt{\ell}\log\lambda}{|\mathcal{G}| - t + 1} \leq \frac{8t\log\lambda}{\sqrt{\ell}} = \epsilon/2.$$

Thus all t sampled indices belong to T with probability at least $1 - \epsilon/2 - \mathrm{neg}(\lambda) \geq 1 - \epsilon$, in which case Sim' outputs $\perp$. Finally, the communication complexity is $O(c\ell) = O(c \cdot t^2 \log^2\lambda/\epsilon^2)$. $\square$

5 ϵ-secure iPIR

In this section we construct a weakly-secure iPIR. We start with our notion of weak security.

Definition 5.1 (Simulator-based ϵ-secure SK-iPIR)**.** *An SK-iPIR is simulator-based ϵ-secure if there exists an interactive PPT simulator* Sim *that, on input $(\widehat{D}, \mathsf{sk})$, has oracle access to an index. Write $\mathsf{Sim}^i(\widehat{D}, \mathsf{sk})$ for the simulator with access to the oracle that returns i. The following two properties are required:*

1. *For every secret key $\mathsf{sk} \leftarrow \mathsf{Gen}(1^\lambda, 1^N)$, database $D \in \{0,1\}^N$, $\widehat{D} \leftarrow \mathsf{Preprocess}(\mathsf{sk}, D)$, index $i \in [N]$, and possibly unbounded interactive server Server^*, during the interaction*

$$\left\langle \mathsf{Sim}^i(\widehat{D}, \mathsf{sk}), \mathsf{Server}^*(\widehat{D}) \right\rangle,$$

   the simulator queries its index oracle with probability at most ϵ.

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

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

   where $\mathsf{sk} \leftarrow \mathsf{Gen}(1^\lambda, 1^N)$ and $\widehat{D} \leftarrow \mathsf{Preprocess}(\mathsf{sk}, D)$. On a query i, the first oracle executes $\mathsf{Client}(\mathsf{sk}, i)$ with $\mathcal{A}$ acting as the server, whereas the second executes $\mathsf{Sim}^i(\widehat{D}, \mathsf{sk})$ with $\mathcal{A}$ acting as the server.

5.1 ϵ-secure iPIR from RKA-security

In the rest of this section we construct ϵ-secure iPIR from batch random-index LPIR. We prove the following lemma.

Lemma 5.2. *Assume the existence of an ϵ/R-secure t-batch RI-LPIR with communication c, for $t = k \log^2 \lambda$, such that $N \geq k^{R-1}$. Then there exists R-round ϵ-secure SK-iPIR with communication $O(R(c + k \cdot \log N) + N/k^{R-1})$.*

We prove Lemma 5.2, but first we derive the following corollary.

Lemma 5.3. *Assume the existence of an XOR-RKA-secure encryption scheme with pseudorandom ciphertexts. Then for any $R, 1/\epsilon \in \mathrm{poly}(N, \lambda)$ there exists R-round ϵ-secure SK-iPIR with communication $R^3 \cdot N^{2/(R+1)}/\epsilon^2 \cdot \mathrm{poly}\lambda$.*

Proof. Let $k = \lceil N^{1/(R+1)} \rceil$ and let $R' \leq R$ be the largest such that $k^{R'} \leq N$. The lemma follows immediately by Lemma 3.3, Lemma 4.2 and Lemma 5.2 with R' rounds, by setting $c = R^2 \cdot N^{2/(R+1)}/\epsilon^2 \cdot \mathrm{poly}(\lambda)$. □

In the following, assume for simplicity that N/k^{R-1} is an integer.[5] Let

$$(\mathsf{RI.Gen}, \mathsf{RI.Preprocess}, \mathsf{RI.Query}, \mathsf{RI.Answer}, \mathsf{RI.Decode})$$

be a t-batch SK-RI-LPIR, used with $t = k \log^2 \lambda$ and ϵ/R security. We next describe the protocol.

[5] Otherwise, pad D to the smallest size $N' \leq 2N$ such that k^{R-1} divides N'.

The iPIR Protocol.

Key generation. $\mathsf{Gen}(1^\lambda, 1^N)$ samples $\mathsf{sk} \leftarrow \mathsf{RI.Gen}(1^\lambda, 1^N)$ and outputs sk.

Preprocessing. $\mathsf{Preprocess}(\mathsf{sk}, D)$ computes $\widehat{D} \leftarrow \mathsf{RI.Preprocess}(\mathsf{sk}, D)$ and outputs $\widehat{D}$.

Query. The query protocol has R rounds. Initially, $D_1 = D$, $M_1 = I_N$, $N_1 = N$, and $i_1 = i$; throughout the protocol, $D_r = M_r D$.

1. At round $r < R$:
   (a) Partition $[N_r]$ into k consecutive blocks $B_1, \ldots, B_k$ of size $N_{r+1} = N_r/k$, view D_r as a $k \times N_{r+1}$ array.
   (b) Client learns a random index $\widehat{j}_\ell$ in each block B_ℓ:
      i. Client and Server interact according to the batch random-index LPIR $\langle \mathsf{RI.Client}(\mathsf{sk}, M_r), \mathsf{RI.Server}(\widehat{D}, M_r)\rangle$ with parameter $t = k \log^2 \lambda$. Client obtains $(j_1, D_r[j_1]), \ldots, (j_t, D_r[j_t])$.
      ii. If there exists $\ell \in [k]$ such that $\{j_1, \ldots, j_t\} \cap B_\ell = \emptyset$, Client aborts. Otherwise, let $\widehat{j}_\ell$ be the first index in $\{j_1, \ldots, j_t\} \cap B_\ell$.
   (c) Let $\ell^* \in [k]$ be such that B_{ℓ^*} contains i_r.
   (d) Client sets shifts $(u_1, \ldots, u_k)$ as follows:
      i. It samples a random shift $\tau \leftarrow \{0, \cdots, N_{r+1} - 1\}$, and sets $u_{\ell^*} = (i_r + \tau) \bmod N_{r+1}$.
      ii. For each other $\ell \in \{1, \ldots, k\} \setminus \{\ell^*\}$, Client sets $u_\ell = (\widehat{j}_\ell + \tau) \bmod N_{r+1}$ and $c_\ell = D_r[\widehat{j}_\ell]$.
      iii. Let $a_r = \bigoplus_{\ell \in \{1,\ldots,k\} \setminus \{\ell^*\}} c_\ell$.
   (e) Client sends $(u_1, \ldots, u_k)$ to Server, and sets $i_{r+1} = \tau$.
   (f) Define
   $$D_{r+1}[v] = \bigoplus_{\ell=1}^{k} D_r\big[\ell, (u_\ell - v) \bmod N_{r+1}\big].$$
   Let M_{r+1} be the corresponding linear map, so that $D_{r+1} = M_{r+1} D$. The construction ensures
   $$D_r[i_r] = a_r \oplus D_{r+1}[i_{r+1}].$$
2. In the final round, Server sends D_R and Client outputs
   $$D_R[i_R] \oplus \bigoplus_{r=1}^{R-1} a_r = D[i].$$

Correctness. We start with proving the correctness and efficiency of the protocol.

Claim 5.4. *The protocol is correct. It has R rounds and communication complexity $R(c + k \cdot \log N) + N/k^{R-1}$.*

Proof. We first observe that each matrix M_r has full rank. Indeed, this can be shown by induction. For $M_1 = I_N$ this is immediate. In round r, we can write

$$M_{r+1} = T_r M_r,$$

where $T_r : \mathbb{F}_2^{N_r} \to \mathbb{F}_2^{N_{r+1}}$ is the linear map defined by

$$T_r x[v] = \bigoplus_{\ell=1}^{k} x\big[\ell, (u_\ell - v) \bmod N_{r+1}\big].$$

We observe that the rows of T_r have disjoint supports: for every block ℓ, and for any two rows $v \neq v' \in [N_{r+1}]$, the coordinates satisfy

$$(u_\ell - v) \bmod N_{r+1} \neq (u_\ell - v') \bmod N_{r+1}.$$

Hence the rows of T_r are linearly independent, and therefore T_r has full row rank. Since, by induction, M_r has full row rank, $M_{r+1} = T_r M_r$ also has full row rank.

For correctness, by the index-uniformity of the random-index LPIR, we may assume that the indices $(j_1, \ldots, j_t)$ in each round are randomly distributed. Indeed, this can be seen by induction over the rounds, where we observe that after replacing the indices $(j_1, \ldots, j_t)$ of the first r rounds with random indices, the matrix M_{r+1} is independent of the secret key, and thus correctness holds with all but negligible probability for round $r + 1$.

Next, fix a round $r < R$. For any block B_ℓ, the probability that none of the $t = k \log^2 \lambda$ uniformly random indices returned by the random-index LPIR lies in B_ℓ is at most

$$\left(1 - \frac{1}{k}\right)^t \leq e^{-t/k} = e^{-\log^2 \lambda}.$$

A union bound over all blocks and rounds, together with the correctness of the underlying random-index LPIR executions, shows that the client aborts or receives an incorrect value only with negligible probability.

Condition on the complementary event. For every $\ell \neq \ell^*$, let x_ℓ be the local coordinate of the index j selected from B_ℓ, and let x_{ℓ^*} be the local coordinate of i_r. By construction, $u_\ell = x_\ell + \tau$ for every ℓ. Hence

$$D_{r+1}[i_{r+1}] = D_{r+1}[\tau] = \bigoplus_{\ell=1}^{k} D_r[\ell, x_\ell] = D_r[i_r] \oplus a_r.$$

Thus $D_r[i_r] = a_r \oplus D_{r+1}[i_{r+1}]$, which gives

$$D[i] = D_R[i_R] \oplus \bigoplus_{r=1}^{R-1} a_r,$$

which proves correctness.

Lastly, notice that for any round r, the last client message of round r (the vector $(u_1, \ldots, u_k)$) can be sent together with the first client message of round $r+1$. Thus the protocol has R communication rounds in total. Each random-index LPIR execution has communication c, and sending the shifts $(u_1, \ldots, u_k)$ costs at most $k \log N$ communication. Moreover, $|D_R| = \frac{N}{k^{R-1}}$.

Overall, the communication is at most $R(c + k \cdot \log N) + \frac{N}{k^{R-1}}$. □

Security. We move to prove security. We define the simulator Sim.

The simulator: $\mathsf{Sim}^i(\mathsf{sk}, \widehat{D})$. Let $\mathsf{RI.Sim}'$ be the simulator of the random-index LPIR. Initially, $D_1 = D$, $M_1 = I_N$, $N_1 = N$, and $i_1 = i$; throughout the protocol, $D_r = M_r D$. Set $k = N^{1/(R+1)}$.

In the following, gray steps are deferred and executed only once the simulator queries the index oracle.

1. At round $r < R$:
   (a) Partition $[N_r]$ into k consecutive blocks $B_1, \ldots, B_k$ of size $N_{r+1} = N_r/k$, view D_r as a $k \times N_{r+1}$ array.
   (b) Execute $\mathsf{RI.Sim}'(\mathsf{sk}, \widehat{D}, M_r)$. If the output of $\mathsf{RI.Sim}'$ is $\perp$:
      i. Sim samples random indices $j'_1, \ldots, j'_t \leftarrow [N_r]$. If there exists $\ell \in [k]$ such that $\{j'_1, \ldots, j'_t\} \cap B_\ell = \emptyset$, Sim aborts.
      ii. Let $\ell^* \in [k]$ be such that B_{ℓ^*} contains i_r.
      iii. Sim samples $(u_1, \ldots, u_k) \leftarrow \{0, \ldots, N_{r+1} - 1\}$ and sends them to Server.
      iv. Let $\tau = (u_{\ell^*} - i_r) \mod N_{r+1}$ and $i'_\ell = (u_\ell - \tau) \mod N_{r+1}$ for every $\ell \neq \ell^*$.
      v. Initialize $(j_1, \ldots, j_t) = (j'_1, \ldots, j'_t)$. Then, for every $\ell \in [k] \setminus \{\ell^*\}$, replace the first element j_s that lies in B_ℓ with $(\ell - 1)N_{r+1} + i'_\ell$.
      vi. Sim sets $i_{r+1} = \tau$.
   (c) Otherwise, let $(j_1, \ldots, j_t)$ be the output of $\mathsf{RI.Sim}'$.
      i. Sim queries its oracle to obtain the index i.
      ii. If there exists $\ell \in [k]$ such that $\{j_1, \ldots, j_t\} \cap B_\ell = \emptyset$, Sim aborts. Otherwise, let $\widehat{j}_\ell$ be the first index in $\{j_1, \ldots, j_t\} \cap B_\ell$.
      iii. Let $\ell^* \in [k]$ be such that B_{ℓ^*} contains i_r.
      iv. Sim sets shifts $(u_1, \ldots, u_k)$ as follows:
         A. It samples a random shift $\tau \leftarrow \{0, \ldots, N_{r+1} - 1\}$ and sets $u_{\ell^*} = (i_r + \tau) \mod N_{r+1}$.
         B. For each other $\ell \in \{1, \ldots, k\} \setminus \{\ell^*\}$, Sim sets $u_\ell = (\widehat{j}_\ell + \tau) \mod N_{r+1}$ and $c_\ell = D_r[\widehat{j}_\ell]$.
      v. Sim sends $(u_1, \ldots, u_k)$ to Server and sets $i_{r+1} = \tau$.
   (d) Define
   $$D_{r+1}[v] = \bigoplus_{\ell=1}^{k} D_r\big[\ell, (u_\ell - v) \mod N_{r+1}\big].$$
   Let M_{r+1} be the corresponding linear map, so that $D_{r+1} = M_{r+1} D$.
2. In the last round, Server sends D_R to Client.

Proof of Lemma 5.2. Correctness, round complexity, and communication complexity follow from Claim 5.4.

For security, define an oracle-aided algorithm $\mathcal{B}$ that, given oracle access to $\mathsf{RI.Client}$, simulates the constructed client for the external server. In every round, $\mathcal{B}$ invokes its oracle on M_r, forwards the messages between the oracle and the server, and then uses the indices returned by the oracle to compute the shifts and continue the protocol. Thus $\mathcal{B}^{\mathsf{RI.Client}}$ is identically distributed to the

real client. By the security of the random-index LPIR, replacing $\mathsf{RI.Client}$ with $\mathsf{RI.Sim}$ produces a computationally indistinguishable execution.

Next, replace the oracle $\mathsf{RI.Sim}$ by $\mathsf{RI.Sim}'$. Whenever $\mathsf{RI.Sim}'$ outputs $\perp$, let $\mathcal{B}$ independently sample $(j_1, \ldots, j_t)$ uniformly at random. By the definition of $\mathsf{RI.Sim}$, this gives exactly the same oracle: $\mathsf{RI.Sim}$ is identical to $\mathsf{RI.Sim}'$, except that it replaces $\perp$ by independent uniform indices.

Finally, consider a round in which $\mathsf{RI.Sim}'$ outputs $\perp$. Sampling the indices $(j_1, \ldots, j_t)$ first and then computing the shift vector $(u_1, \ldots, u_k)$ is identically distributed to first sampling $(u_1, \ldots, u_k)$ uniformly and then sampling $(j_1, \ldots, j_t)$, together with τ, conditioned on producing this shift vector. Indeed, for every fixed current index i_r, the map from the uniform shift τ to u_{ℓ^*} is a bijection, and the local coordinates of the uniform indices in the other blocks are uniform. This reversed sampling order is exactly the behavior of the simulator defined above. Hence its interaction is identically distributed to $\mathcal{B}^{\mathsf{RI.Sim}}$.

The gray steps (of all rounds) are needed only the first time $\mathsf{RI.Sim}'$ does not output $\perp$, and the simulator queries its index oracle and executes these steps only then. This event occurs with probability at most ϵ/R in each round: Fix r, and fix the interaction up to the point before Sim sends the query q_r (this fixes the matrix M_r and the state of Server^*). From this point on, the round-r execution of $\mathsf{RI.Sim}'(\mathsf{sk}, \widehat{D}, M_r)$ is a single session of $\mathsf{RI.Sim}'$ on input M_r, with fresh randomness, and the answer a_r it gets is determined by Server^* and q_r. That is, conditioned on the past, the round-r session interacts with a fixed unbounded server, and by the ϵ/R-security of RI (Definition 4.1, item 1), $\Pr[\mathsf{flag}_r \neq \perp \mid \text{past}] \leq \epsilon/R$. Averaging over the past gives $\Pr[\mathsf{flag}_r \neq \perp] \leq \epsilon/R$. A union bound over the at most R rounds therefore bounds the probability that the simulator queries its index oracle by ϵ, completing the proof. $\square$

6 Security Amplification

In this section we present the security amplification procedure. The amplification procedure is based on Locally Decodable Codes.

Definition 6.1 (Locally Decodable Code)**.** *A q-query non-adaptive locally decodable code (LDC) with message length N and codeword length L consists of a deterministic mapping $C\colon \mathbb{Z}_2^N \to \mathbb{Z}_2^L$ and a randomized oracle algorithm d. On input $\alpha \in [N]$, the decoder first chooses all its queries $\Delta_1, \ldots, \Delta_q \in [L]$, independently of the oracle answers, and then outputs a bit after receiving $C(z)_{\Delta_1}, \ldots, C(z)_{\Delta_q}$. We require that, for every $z \in \mathbb{Z}_2^N$ and $\alpha \in [N]$,*

$$\Pr\left[d^{C(z)}(\alpha) = z_\alpha\right] = 1.$$

The queries are σ-wise input independent *if, for every $S \subseteq [q]$ with $|S| \le \sigma$, there is a distribution $\mathcal{D}_S$ over $[L]^S$ such that, for every input $\alpha \in [N]$, the query tuple generated by $d(\alpha)$ satisfies $(\Delta_j)_{j\in S} \sim \mathcal{D}_S$.*

We say that the code is efficient *if the encoding C is computable in time* $\mathrm{poly}(N, L, q)$*, and d runs in time* $\mathrm{poly}(q, \log L)$*.*

We prove the following amplification lemma:

Lemma 6.2 (Security amplification, general form)**.** *Let W be an R-round ϵ-secure SK-iPIR with communication $c(N, \lambda)$. Let (C, d) be a q-query non-adaptive LDC with σ-wise input-independent queries and codeword length $L = L(N) \le \mathrm{poly}(N, \lambda)$, as in Definition 6.1, such that*

$$\big(\epsilon(L, \lambda) \cdot q\big)^\sigma \le \mathrm{neg}(\lambda).$$

Then there exists a maliciously secure SK-iPIR for databases of length N with $R(L, \lambda)$ rounds and communication $q \cdot c(L, \lambda)$. The protocol runs q executions of W on the encoded database $C(D)$.

The proof is given in Section 6.2, after the description of the protocol. Together with Lemma 5.3, we get the following corollary, which we state as a theorem since it summarizes the main result of this paper. In Section 7 we provide concrete instantiations of the parameters.

Theorem 6.3 (Maliciously secure SK-iPIR from XOR-RKA encryption)**.** *Assume the existence of an XOR-RKA-secure encryption scheme with pseudorandom ciphertexts. Let $N = N(\lambda)$ be a polynomial, and let*

- *$R = R(N) \ge 2$ be a number of rounds,*
- *$\epsilon = \epsilon(N, \lambda) \in (0, 1)$ be an error with $1/\epsilon \le \mathrm{poly}(N, \lambda)$, and*
- *(C, d) be an efficient q-query non-adaptive LDC with σ-wise input-independent queries and codeword length $L = L(N) \le \mathrm{poly}(N, \lambda)$, as in Definition 6.1,*

such that $(\epsilon \cdot q)^\sigma \le \mathrm{neg}(\lambda)$. Then there exists a maliciously secure SK-iPIR for databases of length N with R rounds and

- *communication $q \cdot R^3 \cdot L^{2/(R+1)} \cdot \epsilon^{-2} \cdot \mathrm{poly}(\lambda)$,*

- *client work* $q \cdot R^3 \cdot L^{R/(R+1)} \cdot \epsilon^{-2} \cdot \mathrm{poly}(\lambda) + \mathrm{poly}(q, \log L)$ *per query,*
- *server work* $q \cdot R^2 \cdot L^{1+2/(R+1)} \cdot \epsilon^{-2} \cdot \mathrm{poly}(\lambda)$ *per query, and*
- *server storage* $L \cdot \mathrm{poly}(\lambda)$.

Proof. By Lemma 5.3, applied to databases of length L with R rounds and error ϵ, there is an R-round ϵ-secure SK-iPIR W with communication $R^3 L^{2/(R+1)} \epsilon^{-2} \mathrm{poly}(\lambda)$. The per-query client work, server work and storage of W can be derived by inspection of the protocols of Lemma 3.3, Lemma 4.2 and Lemma 5.2: The storage is the encoding of the database and L keys of λ bits. Let $k = \lceil L^{1/(R+1)} \rceil \geq 2$ and let $R' \leq R$ be the largest such that $k^{R'} \leq L$. Every round $r < R'$ runs one batch SK-RI-LPIR on M_r with $t = k \log^2 \lambda$ and error ϵ/R, which consists of $\ell = 256 t^2 \log^2 \lambda \cdot R^2/\epsilon^2 = O(R^2 k^2 \epsilon^{-2} \log^6 \lambda)$ executions of the SK-1/2LPIR. The rows of M_r have pairwise disjoint supports of size k^{r-1} (by induction on r, since the rows of T_r have pairwise disjoint supports of size k, see the proof of Claim 5.4).

The server derives its folded keys $M_r(k)$ for all rows in time $N_{r-1} \cdot \lambda$ in round r, hence a cost $O(L\lambda)$ overall, and answers each of the ℓ queries of round r by trying its N_r folded keys, for a total of $\sum_r \ell N_r \mathrm{poly}(\lambda) \leq 2\ell L \mathrm{poly}(\lambda) = R^2 L^{1+2/(R+1)} \epsilon^{-2} \mathrm{poly}(\lambda)$.

The client only needs the folded keys $k'_{i,0}$ of the ℓ rows it queries, so it spends $\ell \cdot k^{r-1} \cdot \mathrm{poly}(\lambda)$ time in round r, and work

$$\sum_{r<R'} \ell\, k^{r-1} \mathrm{poly}(\lambda) \leq 2\ell k^{R'-2} \mathrm{poly}(\lambda) = O(R^2 k^{R'} \epsilon^{-2}) \mathrm{poly}(\lambda)$$

(plus the additional low-order cost of reading the final database $D_{R'}$ of size $\leq k^2 \leq k^R$). We finally claim that by our choice of parameters $k^{R'} \leq O(R \cdot L^{R/(R+1)})$ and thus

$$R^2 k^{R'} \epsilon^{-2} \mathrm{poly}(\lambda) \leq R^3 L^{R/(R+1)} \epsilon^{-2} \mathrm{poly}(\lambda).$$

Indeed, $k \leq L^{1/(R+1)} + 1$. When $R < k$, $k^{R'} \leq k^R \leq L^{R/(R+1)}(1 + 1/(k-1))^R \leq e \cdot L^{R/(R+1)}$. On the other hand, when $k \leq R$, by our choice of R', $k^{R'} \leq L \leq L^{1/(R+1)} \cdot L^{R/(R+1)} \leq R \cdot L^{R/(R+1)}$.

The theorem follows by applying Lemma 6.2 to W with the code (C, d). Since the error ϵ of W does not depend on the database length, the hypothesis of the lemma is exactly $(\epsilon q)^\sigma \leq \mathrm{neg}(\lambda)$, and the lemma yields a maliciously secure SK-iPIR with R rounds whose communication, client work and server work are those of W on databases of length L multiplied by q. The client additionally needs to execute d in time $\mathrm{poly}(q, \log L)$. □

6.1 The Amplification Protocol

In the rest of this section we prove Lemma 6.2. We next describe our amplification procedure.

The iPIR Protocol: Let $(\mathsf{W.Gen}, \mathsf{W.Preprocess}, \mathsf{W.Client}, \mathsf{W.Server})$ be an ϵ-secure SK-iPIR. Let (C, d) be a q-query non-adaptive LDC with σ-wise input-independent queries and length $L(N)$.

- **Key Generation:** $\mathsf{Gen}(1^\lambda, 1^N)$ samples $\mathsf{sk} \leftarrow \mathsf{W.Gen}(1^\lambda, 1^{L(N)})$ and outputs sk.
- **Preprocessing:** $\mathsf{Preprocess}(\mathsf{sk}, D)$ given a secret key sk and a database $D \in \{0,1\}^N$, outputs $\mathsf{W.Preprocess}(\mathsf{sk}, C(D))$.

- **Query:** $\langle \mathsf{Client}(\mathsf{sk}, i^*), \mathsf{Server}(\widehat{D})\rangle$: The interactive query protocol between a client Client and a server Server proceeds as follows.
  1. Client runs the first stage of $d(i^*)$ to obtain its query tuple $(i_1, \ldots, i_q)$ and retains the decoder's state.
  2. $\mathsf{Client}, \mathsf{Server}$ interact according to $\langle \mathsf{W.Client}(\mathsf{sk}, i_j), \mathsf{W.Server}(\widehat{D})\rangle$, for every $j \in [q]$, in parallel. Client learns $b_1, \ldots, b_q$.
  3. Client resumes d on answers $(b_1, \ldots, b_q)$ and outputs the resulting bit.

6.2 Security

We start with the security of the protocol. We prove the following lemma; $\epsilon = \epsilon(L, \lambda)$ denotes the error of W on databases of length L.

Lemma 6.4. *Assume that $(\epsilon \cdot q)^\sigma \le \mathrm{neg}(\lambda)$. Then the protocol is maliciously secure.*

Let $\mathsf{W.Client}$ and $\mathsf{W.Server}$ be the client and server of the assumed ϵ-secure SK-iPIR, and let $\mathsf{W.Sim}$ be its simulator. To prove Lemma 6.4, consider an oracle-aided algorithm $\mathcal{B}$ that, given oracle access to $\mathsf{W.Client}$, implements the client of the amplified iPIR protocol. Namely, on input i^*, it runs the first stage of $d(i^*)$ to obtain $(i_1, \ldots, i_q)$ and executes the q oracle sessions on these inputs in parallel. Thus, giving the adversary oracle access to the client of the amplified protocol is identical to giving it oracle access to $\mathcal{B}^{\mathsf{W.Client}}$.

By the security of the weak iPIR, we may replace $\mathsf{W.Client}$ with $\mathsf{W.Sim}$, where in the j-th execution, $\mathcal{B}$ gives $\mathsf{W.Sim}$ oracle access to the index i_j. The goal is to show that, except with negligible probability, at most σ of the q executions of $\mathsf{W.Sim}$ read their input indices.

Claim 6.5. *Let ϵ, q, σ be such that $(\epsilon \cdot q)^\sigma \le \mathrm{neg}(\lambda)$. For each $j \in [q]$, let X_j be the indicator that is 1 when $\mathsf{W.Sim}$ reads its input i_j in the j-th interaction. Then, with all but negligible probability, $\sum X_j \le \sigma$.*

Proof. Fix sk, D and i. Consider a game between $\mathsf{W.Sim}^{\cdot}(\widehat{D}, \mathsf{sk})$ and an (unbounded) algorithm B, where B wins if during the interaction $\mathsf{W.Sim}$ reads i. By the ϵ-security of the protocol, any algorithm B wins this game with probability at most ϵ. By the parallel repetition theorem [Gol99; Gol25], the probability that any algorithm B wins σ parallel games is at most ϵ^σ.

Next, consider q parallel games. We claim that the probability that B wins at least σ games in q parallel games is at most $\binom{q}{\sigma}\epsilon^\sigma \le (\epsilon q)^\sigma$, which is negligible by assumption, for any choice of $(i_1, \ldots, i_q)$. Indeed, fix any subset $S \subseteq [q]$ of σ games. A player B playing only σ games can simulate the view of a player B' playing q games by simulating $\mathsf{W.Sim}$ internally at any other index $i \notin S$. So the probability that B' wins every game in S is at most ϵ^σ. Taking a union bound over the $\binom{q}{\sigma}$ subsets gives the claimed bound.

□

We will make use of the following fact.

Fact 6.6. *Let G be a randomized algorithm that, on input x, outputs $(Y_1, \ldots, Y_q) \in \Omega^q$. Suppose its outputs are σ-wise input independent: for every $S \subseteq [q]$ with $|S| \le \sigma$, there is a distribution $\mathcal{D}_S$ such that $(Y_j)_{j\in S} \sim \mathcal{D}_S$ for every input x. Let $\mathcal{A}$ be an oracle-aided algorithm that accesses its oracle in at most σ locations. Then the distribution of the view of $\mathcal{A}^{(Y_1,\ldots,Y_q)}$, for $(Y_1, \ldots, Y_q) \leftarrow G(x)$, is independent of x.*

Proving Lemma 6.4

Proof of Lemma 6.4. By the security of the weak iPIR, having an oracle to $\mathcal{B}^{\mathsf{W.Client}}$ is indistinguishable from having an oracle to $\mathcal{B}^{\mathsf{W.Sim}}$. We can equivalently rewrite the latter oracle as $\mathcal{B}'^{\mathsf{W.Sim}}$, where, on input i^*, $\mathcal{B}'$ first runs the first stage of $d(i^*)$ to obtain

$$(i_1, \ldots, i_q),$$

and then runs an algorithm $\mathcal{C}^{\mathsf{W.Sim},(i_1,\ldots,i_q)}$ that simulates the remainder of $\mathcal{B}$. The algorithm $\mathcal{C}$ accesses a coordinate i_j only if the corresponding execution of $\mathsf{W.Sim}$ queries its index oracle.

Truncate $\mathcal{C}$ just before it reads a $(\sigma+1)$-st distinct coordinate. By Claim 6.5, this changes its view only with negligible probability. The truncated algorithm accesses i^* only through the query vector generated by the first stage of d, so Fact 6.6 implies that its view is identical when this vector is generated on input i^* or on input 1 (note that the LDC decoder state is used only after the parallel executions to compute the client's local output and is not part of the server's view). Thus, an oracle to $\mathcal{B}'^{\mathsf{W.Sim}}$ is indistinguishable from an oracle to the algorithm $\mathcal{B}''$ that ignores its input and samples its queries according to $d(1)$. The resulting distribution is independent of i^*, and thus the real client oracle is indistinguishable from an oracle whose behavior is independent of i^*. □

Proving Lemma 6.2

Proof of Lemma 6.2. The protocol runs the first stage of $d(i^*)$ to obtain indices $(i_1, \ldots, i_q)$ and then runs q executions of W in parallel on the database $C(D) \in \{0,1\}^L$ and these indices. Since $L \le \mathrm{poly}(N, \lambda)$ is polynomial in λ, the correctness, the ϵ-security and the complexity bounds of W apply to databases of length L. The q executions are synchronized round by round, so the protocol has $R(L,\lambda)$ rounds and communication $q \cdot c(L,\lambda)$. The server stores $\mathsf{W.Preprocess}(\mathsf{sk}, C(D))$, of length $s(L,\lambda)$, and runs the server of W q times, for a total work of $q \cdot w_{\mathsf{Server}}(L,\lambda)$. The client runs d once and runs the client of W q times, for a total work of $q \cdot w_{\mathsf{Client}}(L,\lambda)$ plus the running time of d. By a union bound over the q executions, all the outputs $b_j = C(D)_{i_j}$ are correct except with probability q times the correctness error of W; conditioned on this event, resuming d on $(b_1, \ldots, b_q)$ outputs $D[i^*]$ by the perfect correctness of the code. Finally, the hypothesis $(\epsilon(L,\lambda) \cdot q)^\sigma \le \mathrm{neg}(\lambda)$ is that of Lemma 6.4, which gives malicious security. □

7 Instantiations

In this section we provide the concrete instantiations of the parameters which we used to derive the numbers reported in Table 2. We will need the following lemma, proven for completeness in Section A.

Lemma 7.1 ([GS92])**.** *Fix integers $\sigma, w, r, N > 0$ such that $N \le \binom{r+w}{r}$. There exists a q-query non-adaptive LDC (C,d) with message length N, codeword length L, and σ-wise input-independent queries, where $q = O\big(r\sigma \log(r\sigma+2)\big)$ and $L \le \big(4^{w+1}\sigma^{w+1}r^{w+1}\big)$.*

The encoding C is computable in time $\mathrm{poly}(N, L, q)$, and the query sampler d runs in time $\mathrm{poly}(q, \log L)$.

We prove the following corollary.

Corollary 7.2 (Instantiations)**.** *Assume the existence of an XOR-RKA-secure encryption scheme with pseudorandom ciphertexts, and let $N = N(\lambda)$ be a polynomial. There exist maliciously secure SK-iPIR protocols for databases of length N with the following parameters, where all bounds hide* $\mathrm{poly}(\lambda)$ *factors.*

1. (Polylogarithmic communication.) *For every constant $\delta > 0$: $O(\log N)$ rounds, communication $\log^{3/\delta+O(1)} N$, client work and server work $N^{1+\delta+o(1)}$, and server storage $N^{1+\delta+o(1)}$.*

2. (Round-communication trade-off.) *For every $R = R(N)$ with $2 \le R \le \log N$: R rounds, communication $N^{2/(R+1)+o(1)}$, client work $N^{1-1/(R+1)+o(1)}$, server work $N^{1+2/(R+1)+o(1)}$, and server storage $N^{1+o(1)}$, where all $o(1)$ terms are $O(\sqrt{\log\log N/\log N})$. In particular:*

   - *for every $R = \omega(1)$, the communication is $N^{o(1)}$ and both the client work and the server work are $N^{1+o(1)}$;*
   - *for every constant R, writing $\varepsilon = 2/(R+1)$, the communication is $N^{\varepsilon+o(1)}$, the client work is $N^{1-\varepsilon/2+o(1)}$ and the server work is $N^{1+\varepsilon+o(1)}$.*

Proof. Assume that $N > \lambda$, as otherwise the trivial protocol in which the server sends D has the desired properties. Both items use Theorem 6.3 with the code of Lemma 7.1 for $\sigma = \lceil \log\log N \rceil$, and with $\epsilon = 1/(q\lambda)$. Then $1/\epsilon = \mathrm{poly}(N, \lambda)$, $\epsilon^{-2} = q^2\lambda^2$, and $(\epsilon q)^\sigma = \lambda^{-\sigma} = \mathrm{neg}(\lambda)$ since $\sigma = \omega(1)$.

Polylogarithmic communication. For item 1, let $c = 1/\delta + 1$, and let $r = \lceil \log^c N \rceil$ and $w = \lceil \log N/((c-1)\log\log N) \rceil$. Since $w \le \log N$, we have $\log(r/w) \ge (c-1)\log\log N$, hence $\binom{r+w}{w} \ge (r/w)^w \ge N$. Lemma 7.1 gives $q = O(r\sigma\log(r\sigma+2)) = O(\log^c N\,(\log\log N)^2)$ and

$$\begin{aligned}\log L &\le (w+1)\log(4\sigma r) + O(1)\\ &= \Big(\frac{\log N}{(c-1)\log\log N} + O(1)\Big)\big(c\log\log N + O(\log\log\log N)\big)\\ &= \Big(1 + \frac{1}{c-1} + o(1)\Big)\log N.\end{aligned}$$

Take $R = \lceil \log L \rceil = O(\log N)$, so that $L^{2/(R+1)} \le 4$ and $L^{R/(R+1)} \le L$. Theorem 6.3 gives communication $q^3R^3L^{2/(R+1)}\mathrm{poly}(\lambda) = \log^{3c+O(1)} N \cdot \mathrm{poly}(\lambda)$, and client and server work at most $4q^3R^2L \cdot \mathrm{poly}(\lambda) = L \cdot \mathrm{polylog}(N) \cdot \mathrm{poly}(\lambda) = N^{1+1/(c-1)+o(1)}\mathrm{poly}(\lambda)$. The storage is $L \cdot \mathrm{poly}(\lambda)$.

Round-communication trade-off. For item 2, let $w = \lceil \sqrt{\log N/\log\log N} \rceil$ and $r = \lceil wN^{1/w} \rceil$. Then $\binom{r+w}{w} \ge (r/w)^w \ge N$, $q = O(r\sigma\log(r\sigma+2)) = 2^{O(\sqrt{\log N\log\log N})} = N^{O(\sqrt{\log\log N/\log N})}$, and $\log L \le (w+1)(\log(4\sigma) + \log r) + O(1) = \log N + O(\sqrt{\log N\log\log N})$, that is, $L = N^{1+O(\sqrt{\log\log N/\log N})}$. The bounds follow by plugging q, $\epsilon^{-2} = q^2\lambda^2$ and L into Theorem 6.3, using $R^3 \le \log^3 N$, and observing that $O(\sqrt{\log\log N/\log N}) = o(1)$. □

AI Usage

We have relied on OpenAI's Prism (5.6 Sol, Astra) for editing, formatting, and figure generation.

Acknowledgments

Nir Bitansky was partially supported by the Advanced Research and Invention Agency (ARIA). Geoffroy Couteau was supported by the French Agence Nationale de la Recherche (ANR) through the France 2030 project SecureCompute (ANR-22-PECY-003), and by the European Research Council (ERC) through grant OBELiSC (101115790). Noam Mazor was partially supported by the Simons Foundation.

References

[ABG26] Benny Applebaum, Nir Bitansky, and Nathan Geier. "Security Amplification via Robust Indistinguishability Combiners". In: *Advances in Cryptology—CRYPTO 2026, Part I*. Springer, 2026, pp. 35–66. DOI: 10.1007/978-3-032-35367-2_2 (cit. on p. 6).

[AHI11] Benny Applebaum, Danny Harnik, and Yuval Ishai. "Semantic Security under Related-Key Attacks and Applications". In: *Proceedings of the 2nd Symposium on Innovations in Computer Science*. Tsinghua University Press, 2011, pp. 45–60 (cit. on pp. 2, 5, 13).

[App13] Benny Applebaum. "Garbling XOR Gates "For Free" in the Standard Model". In: *Theory of Cryptography*. Vol. 7785. Lecture Notes in Computer Science. Springer, 2013, pp. 162–181. DOI: 10.1007/978-3-642-36594-2_10 (cit. on p. 5).

[BCM22] Elette Boyle, Geoffroy Couteau, and Pierre Meyer. "Sublinear secure computation from new assumptions". In: *Theory of Cryptography Conference*. Springer. 2022, pp. 121–150 (cit. on pp. 1, 4).

[Ben+25] Fabrice Benhamouda, Caicai Chen, Shai Halevi, Yuval Ishai, Hugo Krawczyk, Tamer Mour, Tal Rabin, and Alon Rosen. "Encrypted Matrix-Vector Products from Secret Dual Codes". In: *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security*. 2025. DOI: 10.1145/3719027.3765194 (cit. on p. 9).

[BGIK22] Elette Boyle, Niv Gilboa, Yuval Ishai, and Victor I Kolobov. "Programmable distributed point functions". In: *Annual International Cryptology Conference*. Springer. 2022, pp. 121–151 (cit. on p. 6).

[BIM04] Amos Beimel, Yuval Ishai, and Tal Malkin. "Reducing the Servers' Computation in Private Information Retrieval: PIR with Preprocessing." In: *Journal of cryptology* 17.2 (2004) (cit. on pp. 1, 8).

[BIP18] Elette Boyle, Yuval Ishai, and Antigoni Polychroniadou. "Limits of Practical Sublinear Secure Computation". In: *Advances in Cryptology—CRYPTO 2018*. Vol. 10993. Lecture Notes in Computer Science. Springer, 2018, pp. 302–332. DOI: 10.1007/978-3-319-96878-0_11 (cit. on pp. 3, 4, 9).

[BIPW17] Elette Boyle, Yuval Ishai, Rafael Pass, and Mary Wootters. "Can we access a database both locally and privately?" In: *Theory of Cryptography Conference*. Springer. 2017, pp. 662–693 (cit. on pp. 1, 8, 9).

[BK03] Mihir Bellare and Tadayoshi Kohno. "A theoretical treatment of related-key attacks: RKA-PRPs, RKA-PRFs, and applications". In: *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2003, pp. 491–506 (cit. on pp. 2, 5).

[BM26] Nir Bitansky and Noam Mazor. "Secret-Key PIR from One-Way Functions". In: *Theory of Cryptography Conference*. Cryptology ePrint Archive, Paper 2026/865. 2026 (cit. on pp. 1, 2, 8).

[CGKS98] Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. "Private information retrieval". In: *Journal of the ACM (JACM)* 45.6 (1998), pp. 965–981 (cit. on pp. 1, 8).

[Che+26] Caicai Chen, Yuval Ishai, Aayush Jain, Tamer Mour, Alon Rosen, and Chaoping Xing. *Doubly-Efficient Secret-Key PIR with Low Storage Overhead*. Cryptology ePrint Archive, Paper 2026/1480. 2026. URL: https://eprint.iacr.org/2026/1480 (cit. on p. 9).

[CHR17] Ran Canetti, Justin Holmgren, and Silas Richelson. "Towards doubly efficient private information retrieval". In: *Theory of Cryptography Conference*. Springer. 2017, pp. 694–726 (cit. on pp. 8, 9).

[CIMR26] Caicai Chen, Yuval Ishai, Tamer Mour, and Alon Rosen. "Secret-key PIR from random linear codes". In: *Proceedings of the 58th Annual ACM Symposium on Theory of Computing*. 2026 (cit. on pp. 1, 2, 8, 9).

[DMO00] Giovanni Di Crescenzo, Tal Malkin, and Rafail Ostrovsky. "Single Database Private Information Retrieval Implies Oblivious Transfer". In: *International Conference on the Theory and Applications of Cryptographic Techniques*. 2000 (cit. on pp. 1, 8).

[GGM84] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. "On the Cryptographic Applications of Random Functions". In: *Advances in Cryptology: Proceedings of CRYPTO 84*. 1984, pp. 276–288 (cit. on p. 13).

[GHMNY21] Craig Gentry, Shai Halevi, Bernardo Magri, Jesper Buus Nielsen, and Sophia Yakoubov. "Random-index PIR and applications". In: *Theory of Cryptography Conference*. Springer. 2021, pp. 32–61 (cit. on pp. 4, 7).

[Gol25] Oded Goldreich. "On parallel repetition of interactive proof systems". In: *Computational Complexity and Local Algorithms: On the Interplay Between Randomness and Computation*. Springer, 2025, pp. 119–126 (cit. on pp. 7, 32).

[Gol99] Oded Goldreich. *Modern cryptography, probabilistic proofs and pseudorandomness*. Vol. 17. Springer, 1999 (cit. on p. 32).

[GS92] Peter Gemmell and Madhu Sudan. "Highly resilient correctors for polynomials". In: *Information processing letters* 43.4 (1992), pp. 169–174 (cit. on p. 33).

[GZSP25] Ashrujit Ghoshal, Mingxun Zhou, Elaine Shi, and Bo Peng. "Pseudorandom Functions with Weak Programming Privacy and Applications to Private Information Retrieval". In: *Advances in Cryptology—EUROCRYPT 2025*. Vol. 15607. Lecture Notes in Computer Science. Full version available as Cryptology ePrint Archive, Paper 2025/300. Springer, 2025, pp. 284–313. DOI: 10.1007/978-3-031-91098-2_11. URL: https://eprint.iacr.org/2025/300 (cit. on p. 6).

[HILL99] Johan Hastad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. "A pseudorandom generator from any one-way function". In: *SIAM Journal on Computing* (1999), pp. 1364–1396 (cit. on p. 13).

[IR89] Russell Impagliazzo and Steven Rudich. "Limits on the provable consequences of one-way permutations". In: *Proceedings of the 21st Annual ACM Symposium on Theory of Computing (STOC)*. ACM Press, 1989, pp. 44–61 (cit. on p. 1).

[Kil92] Joe Kilian. "A Note on Efficient Zero-Knowledge Proofs and Arguments (Extended Abstract)". In: *Proceedings of the 24th Annual ACM Symposium on Theory of Computing*. 1992, pp. 723–732 (cit. on p. 6).

[KO00] Eyal Kushilevitz and Rafail Ostrovsky. "One-Way Trapdoor Permutations Are Sufficient for Non-trivial Single-Server Private Information Retrieval". In: *Advances in Cryptology—EUROCRYPT 2000*. Vol. 1807. Lecture Notes in Computer Science. Springer, 2000, pp. 104–121 (cit. on p. 1).

[KO97] Eyal Kushilevitz and Rafail Ostrovsky. "Replication is NOT Needed: SINGLE Database, Computationally-Private Information Retrieval". In: *Proceedings 38th annual symposium on foundations of computer science*. 1997, pp. 364–373 (cit. on pp. 1, 8).

[KS08] Vladimir Kolesnikov and Thomas Schneider. "Improved Garbled Circuit: Free XOR Gates and Applications". In: *Automata, Languages and Programming*. Vol. 5126. Lecture Notes in Computer Science. Springer, 2008, pp. 486–498. DOI: 10.1007/978-3-540-70583-3_40 (cit. on pp. 4, 5, 12).

[KT00] Jonathan Katz and Luca Trevisan. "On the efficiency of local decoding procedures for error-correcting codes". In: *Proceedings of the thirty-second annual ACM symposium on Theory of computing*. 2000, pp. 80–86 (cit. on p. 8).

[LMW23] Wei-Kai Lin, Ethan Mook, and Daniel Wichs. "Doubly efficient private information retrieval and fully homomorphic RAM computation from ring LWE". In: *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*. 2023, pp. 595–608 (cit. on p. 8).

[Mic00] Silvio Micali. "Computationally Sound Proofs." In: *SIAM Journal on Computing* (2000). Preliminary version in *FOCS'94*, pp. 1253–1298 (cit. on p. 6).

A Proving Lemma 7.1

In this section we prove Lemma 7.1, restated below.

Lemma A.1 (Lemma 7.1, restated)**.** *Fix integers $\sigma, w, r, N > 0$ such that $N \le \binom{r+w}{r}$. There exists a q-query non-adaptive LDC (C, d) with message length N, codeword length L, and σ-wise input-independent queries, where $q = O\big(r\sigma \log(r\sigma + 2)\big)$ and $L \le \big(4^{w+1}\sigma^{w+1}r^{w+1}\big)$.*

The encoding C is computable in time $\mathrm{poly}(N, L, q)$, and the query sampler d runs in time $\mathrm{poly}(q, \log L)$.

Proof. Let $Q = 2^m$ be the smallest power of two strictly larger than $r\sigma + 1$, fix $\mathbb{F} = \mathbb{F}_Q$. Thus $r\sigma + 1 < |\mathbb{F}| \le 2r\sigma + 2$. Let $N' = \lceil N/m \rceil$. By padding with zeros, we view $z \in \mathbb{F}_2^N$ as an element of $\mathbb{F}^{N'}$ using a fixed $\mathbb{F}_2$-basis of $\mathbb{F}$. By assumption

$$Q^w \ge (r+1)^w \ge \binom{r+w}{w} \ge N \ge N'.$$

Let $M = \binom{r+w}{w}$. Fix distinct elements $x_1, \dots, x_M \in \mathbb{F}^w$ such that (1) the mapping $i \mapsto x_i$ is computable in time $\mathrm{poly}(q, \log L)$, and (2), for every choice of $z_1, \dots, z_M \in \mathbb{F}$, there exists a polynomial P_z of degree at most r such that $P_z(x_i) = z_i$ for all $i \in [M]$.

$x_1, \dots, x_M$ can be defined as follows. Fix distinct $b_0, \dots, b_r \in \mathbb{F}$, and order the set

$$\mathcal{I} = \{\beta \in \mathbb{Z}_{\ge 0}^w : \beta_1 + \cdots + \beta_w \le r\}$$

by total degree and then lexicographically. Write its elements as $\beta^{(1)}, \dots, \beta^{(M)}$, and set $x_i = (b_{\beta_1^{(i)}}, \dots, b_{\beta_w^{(i)}})$ for $i \in [M]$. The polynomials

$$B_\beta(X) = \prod_{j=1}^{w} \prod_{h=0}^{\beta_j - 1} (X_j - b_h)$$

have total degree at most r, satisfy $B_{\beta^{(i)}}(x_i) \ne 0$, and satisfy $B_{\beta^{(i)}}(x_k) = 0$ for $k < i$: an earlier multi-index has some coordinate smaller than the corresponding coordinate of $\beta^{(i)}$. Thus every assignment of values at these M points can be written as a linear combination of the above polynomials.

Encoding. Given $z \in \mathbb{F}^{N'}$, find a polynomial $P_z \in \mathbb{F}[X_1, \dots, X_w]$ of total degree at most r such that $P_z(x_i) = z_i$ for every $i \in [N']$, and set its value to zero on the remaining interpolation points. The codeword consists of the field evaluation

$$C(z)_x = P_z(x) \qquad \text{for every } x \in \mathbb{F}^w.$$

Thus the codeword has Q^w field symbols, or binary length

$$L = mQ^w \le Q^{w+1} \le (2r\sigma + 2)^{w+1} \le 4^{w+1}r^{w+1}\sigma^{w+1}.$$

Query sampling. Set $D = r\sigma$. Fix distinct nonzero $a_1, \ldots, a_{D+1} \in \mathbb{F}$. By Lagrange interpolation, there exist fixed coefficients $c_1, \ldots, c_{D+1} \in \mathbb{F}$ such that every polynomial $f \in \mathbb{F}[X]$ of degree at most D satisfies

$$f(0) = \sum_{\ell=1}^{D+1} c_\ell f(a_\ell).$$

On input $\alpha \in [N]$, let $\bar{\alpha} = \lceil \alpha/m \rceil$ be the field-symbol index containing the desired bit, and let $t_\alpha = \alpha - (\bar{\alpha} - 1)m$ be its position within that symbol. Sample independent uniform $v_1, \ldots, v_\sigma \in \mathbb{F}^w$ and define

$$\gamma(T) = x_{\bar{\alpha}} + \sum_{h=1}^{\sigma} v_h T^h.$$

For every $\ell \in [D+1]$, the decoder reads the entire field element $P_z(\gamma(a_\ell))$, namely all m bits of the codeword block indexed by $\gamma(a_\ell)$. After receiving these values, it computes

$$y = \sum_{\ell=1}^{D+1} c_\ell P_z(\gamma(a_\ell))$$

and outputs the t_α-th bit of y. The number of binary queries is

$$q = (D+1)m = O\big(r\sigma \log(r\sigma + 2)\big).$$

Correctness. The polynomial $f(T) = P_z(\gamma(T))$ has degree at most D. By the interpolation identity,

$$y = \sum_{\ell=1}^{D+1} c_\ell f(a_\ell) = f(0) = P_z(x_{\bar{\alpha}}) = z_{\bar{\alpha}}.$$

Therefore, the t_α-th bit of y is exactly the original message bit z_α.

Input independence. Fix a set S of at most σ positions in the binary query tuple, and let $J \subseteq [D+1]$ be the set of curve evaluations touched by these positions. Since $|J| \le \sigma$, the points $(\gamma(a_\ell))_{\ell \in J}$ are independent uniform elements of $\mathbb{F}^w$, regardless of α. Each query in S consists of one of these points and a fixed bit position in $[m]$, so its joint distribution is independent of α.

Running time. Computing the mapping $i \mapsto x_i$ can be done efficiently by computing the binomial coefficients $\binom{s+t}{t}$ for every $s \le r, t \le w$ in time $\mathrm{poly}(w, r, \log M) \le \mathrm{poly}(q, \log L)$. The encoding can be computed in time $\mathrm{poly}(N, L, q)$ by padding the field-symbol message with zeros to length M, solving a linear system to find P_z, and evaluating P_z on every point of $\mathbb{F}^w$.

For the decoding, given α, sampling the curve and evaluating it at all a_ℓ takes $O(Dw\sigma)$ field operations. Constructing the field and computing the interpolation coefficients takes $\mathrm{poly}(q)$ time. Reading the $D+1$ field elements and reconstructing y takes $O(q)$ bit operations plus $O(D)$ field operations, so the decoder runs in $\mathrm{poly}(q, \log L)$ time.

□

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il