Streamlined Symmetric Private Information Retrieval via Rényi Divergence

Alex Davidson* Nuno Nogueira†* Samuel Pearson‡ João Ribeiro‡

May 25, 2026

Abstract

Private Information Retrieval (PIR) protocols allow a client to recover items from a server-held database without revealing the locations of requested items. In Symmetric PIR (SPIR), the client also learns nothing about the database beyond the requested items. Such schemes are critical for maintaining security in applications such as compromised credential checking, where database elements are considered as sensitive as queries. Existing approaches to building SPIR schemes require running multiple cryptographic primitives in parallel. Moreover, they do not naturally translate to the post-quantum setting, even though practical PIR schemes are typically post-quantum due to their reliance on learning with errors (LWE).

This work explores the possibility of deriving SPIR from PIR directly, utilising noise flooding to maintain the privacy of the database. While the common analysis based on the statistical distance leads to impractical parameters, we instead utilise arguments based on the Rényi divergence to obtain significantly improved parameters. We obtain simple single-server SPIR from state-of-the-art LWE-based PIR schemes with polynomial noise dimension and ciphertext modulus (concretely of 64 bits in size). Along the way, we note that practical schemes that utilise preprocessing via client-downloaded offline hints require extra protections for the database.

Overall, via an implementation of our approach, we show that post-quantum, round-optimal SPIR schemes can be constructed requiring online communication of 8 MB and server computation costs of 302 ms for a database of 1 million 1 kB elements.

1 Introduction

Private information retrieval (PIR), introduced by Chor, Goldreich, Kushilevitz, and Sudan [14], enables clients to access entries from a server-held database while ensuring that the positions of queried entries remain private. Practical applications of PIR include services such as URL filtering [6, 34], password checking [5], and privacy-preserving advertising [53, 47, 28]. Gertner, Ishai, Kushilevitz, and Malkin [26] proposed *symmetric* PIR (SPIR), which has the additional requirement that clients learn nothing about database entries that are not queried. In general, SPIR considers

*LASIGE, Faculdade de Ciências, Universidade de Lisboa. alex.davidson@fc.ul.pt. Funded by national funds through FCT — Fundação para a Ciência e a Tecnologia, I.P., under the project "Fully-Homomorphic Encryption from Post-Quantum Code-Based Assumptions", ref. 2024.12595.CMU, DOI 10.54499/2024.12595.CMU, and under the LASIGE Research Unit, ref. UID/00408/2025, DOI 10.54499/2024.07643.IACDC.

†NOVA FCT, Universidade Nova de Lisboa. nm.nogueira@campus.fct.unl.pt.

‡Instituto de Telecomunicações and Departamento de Matemática, Instituto Superior Técnico, Universidade de Lisboa. {samuelpearson,jribeiro}@tecnico.ulisboa.pt. Funded by national funds through FCT – Fundação para a Ciência e a Tecnologia, I.P., and, when eligible, co-funded by EU funds under project/support UID/50008/2025 – Instituto de Telecomunicações, with DOI 10.54499/UID/50008/2025.

applications and settings where database items are considered as sensitive as the queries that are being made. Such applications include pay-per-access platforms, such as streaming services or event tickets, where the server wishes to charge the client for downloaded information, and the client wants to keep private the content they are streaming; and queries of sensitive data, such as medical, legal, or financial records.[1].

Practical PIR. While original PIR work focused on information-theoretic privacy, which requires multiple non-colluding servers, there has been significant recent focus on the single-server setting, where the database is owned by one server and query privacy is based solely on computational hardness assumptions [35]. To drive down the computational costs of PIR, Beimel, Ishai, and Malkin were the first to consider the preprocessing model (in the multi-server setting) [8], in which the servers perform some query-independent computation before answering queries from clients. A more recent line of work in single-server PIR also moves large proportions of expensive computation, as well as communication, to an offline phase (e.g., [50, 18, 17, 20, 32, 58]). Such schemes are known as *stateful*, and are able to obtain significant improvements in the amortized costs of answering online queries.

In particular, the family of LWE-based schemes initiated with FrodoPIR [20] and SimplePIR [32] are able to obtain very high server throughput (close to memory bandwidth), with the main drawbacks being per-query communication, client-side offline download, and server-side preprocessing (though the preprocessing costs amortize across *all queries* from *all clients*). Subsequent works have targeted the two latter costs [31, 38, 44, 42]. Despite having suboptimal asymptotics, these schemes are extremely competitive from a practical standpoint, particularly in the scenario of many queries to a static database, and have extremely simple and versatile designs (for instance, they are compatible with efficient reductions from PIR by keywords [15, 13, 29]).

Existing symmetric PIR schemes. While PIR continues to attract significant attention, especially in the past decade when performance advancements have led to real-world deployment (e.g. [54]), single-server SPIR research has been comparably scarce. In fact, the black-box reduction from SPIR to PIR of Freedman, Ishai, Pinkas, and Reingold [23] via the use of Oblivious Pseudorandom Functions (OPRFs) [24] has stood as a reference for obtaining single-server SPIR "for free". The idea behind the reduction is to encrypt each database element in position i under $\mathsf{F}_{\mathsf{sk}}(i)$, where F is a pseudorandom function and sk is a server-held secret key. This allows the client to privately learn the values they need, and nothing else due to the encryption. Thus, the standard approach to obtaining symmetric privacy requires running multiple cryptographic primitives in parallel. Although this reduction is efficient by making use of efficient classical OPRF constructions, post-quantum candidates for OPRFs perform significantly worse [3, 4, 30, 19]. Therefore, achieving post-quantum secure SPIR would appear to be currently out of reach (even though practical, LWE-based PIR schemes are quantum-safe by design).[2]

Our goal. We aim to achieve SPIR with as little performance separation from PIR as possible, while retaining simplicity. Informally, we would like to say that a SPIR query is, operationally and conceptually, just a single query, rather than a PIR query augmented with an auxiliary cryptographic primitive such as an OPRF. The central question we address is the following:

[1] We note that, in such cases it may be crucial to add an authentication mechanism to also limit *who* can access *what* [37, 33, 57]

[2] Requiring an expensive and interactive offline phase for each client, which drives up costs significantly for global deployments, see Section 8 for a discussion.

Can we construct *simple* single-server SPIR protocols with *concrete performance* comparable to state-of-the-art LWE-based PIR protocols?

1.1 Our Contributions

In this work, we present SPIR schemes that build on LWE-based PIR by using noise flooding techniques backed by an analysis based on Rényi divergence. Our main scheme is constructed from FrodoPIR [20], and we also construct a scheme based on the recursive variant of SimplePIR known as DoublePIR [32]. As is standard in PIR (and SPIR), we assume both parties to be semi-honest[3], and our main scheme considers clients who query the database adaptively.

The main performance hits taken to achieve database privacy are due to the increase in ciphertext modulus (and, consequently, LWE dimension) to allow space for the flooding noise, and due to the extra work, storage and communication needed to encode each database element in the offline phase and to decode elements obtained in online query responses. However, this encoding/decoding process is computationally cheap (it essentially consists of evaluating an efficient hash function per element), and the main overhead comes from the memory and bandwidth costs.

Noise flooding and Rényi divergence for improved security proofs. A major technical aspect of our work is the use of the Rényi divergence in our security proofs in order to obtain better parameters. The Rényi divergence was first used in lattice-based cryptography in the decision-to-search reduction for the Ring-LWE problem [41], which has since become ubiquitous. The first extensive study on the use of the Rényi divergence, rather than the statistical distance, as a measure of distribution closeness in security proofs for lattice-based cryptography was given by Bai, Lepoint, Roux-Langlois, Sakzad, Stehlé, and Steinfeld [7]. This work identified a property of decision problems called *public sampleability* for which the Rényi divergence could be used in security proofs in a modular way (this framework has recently been expanded [9]). However, as we later explain, our case does not seem to fit into this framework. The Rényi divergence has found many applications in security proofs, such as for multilinear maps, threshold PKE, FHE, and signature schemes [36, 11, 16, 1].

We note that, in the FrodoPIR/SimplePIR family of LWE-based PIR protocols, query responses contain an error term which is obtained by multiplying the small error from the client's LWE-encrypted query by a matrix which encodes the database. This leakage can be statistically hidden by simply adding sufficiently large random noise to the response - a technique known as *noise flooding*. The effect of noise flooding on the scheme's parameters directly impacts the ciphertext expansion factor, since ciphertexts have to accommodate larger error terms while enabling correct decryption. Moreover, since this increases the LWE modulus-to-noise ratio, the LWE dimension also has to increase to compensate for the security loss. The overall structure of a SPIR scheme obtained via this method, however, is virtually identical to the underlying PIR scheme. Our key contribution is to analyze noise flooding with the Rényi divergence, as a way of minimizing the parameter blow-up. We use this technique to analyze our first SPIR scheme, presented in Section 4. We note that, along the way, we must develop more appropriate database privacy notions to fit our approach, which we describe in the following paragraph.

[3] In fact, our security proofs still hold if clients are allowed to choose their random tape arbitrarily, but otherwise follow the protocol.

New database privacy notions. Due to our use of the Rényi divergence, we avoid the standard definition of database privacy based on statistical indistinguishability [4], and instead opt for a game-based approach. For technical reasons which we discuss in the next section, it is useful in first instance to define a database privacy notion based on a *search* problem. We call this notion *one-way database privacy*, which is defined as the hardness of guessing the value of any database element which is not queried. Clearly, this definition requires the assumption that database elements are hard to guess *a priori*. Thus, we consider database privacy to be parameterized by a class of prior database distributions for which the aforementioned search problem is hard.

This first privacy notion is obviously weak. In particular, it does not preclude the possibility of an adversary gaining sensitive partial information from the database, even if they are not able to reconstruct an *entire* unqueried element. However, it is worth noting that one-way database privacy may be sufficient in certain applications, for instance when clients pay to retrieve a token that gives them access to some product.

We present a stronger notion called *indistinguishability-based database privacy*, which requires that the *posterior* distribution of any unqueried element (i.e., their distribution after the adversary has made some bounded number of queries to the database) remains indistinguishable from its prior distribution. We then show how to achieve this stronger privacy notion (in the random oracle model) for *any prior database distribution* by appropriately transforming a scheme which only satisfies the weaker notion for the *uniform* prior distribution. We additionally show how to achieve the stronger privacy notion in the standard model, but only for the uniform prior distribution. These transformations are given in Section 5.

We consider two flavors of both these definitions of database privacy: selective and adaptive. In the selective case, the adversary must choose all their query indices at once, while in the adaptive case the next query index can depend on previous responses. Our formal database privacy definitions are given in Section 3. Finally, we provide a (naive) reduction from adaptive to selective database privacy which simplifies our security proofs.

Concrete parameters. As mentioned, not only is our ciphertext modulus polynomial in the security parameter, which represents an asymptotic improvement relative to the standard noise flooding approach with a statistical distance analysis, but it is *concretely* only a factor of $2\times$ larger than the 32-bit modulus used in FrodoPIR and SimplePIR. Since it appears infeasible to maintain the same modulus after noise flooding, and given that a power-of-two modulus is desirable for practical purposes, this is essentially the best possible outcome. Moreover, we note that 64-bit elements fit natively in the ALU registers of common hardware, which is crucial for keeping the practical cost of operations small. When considering 1KB-sized elements, the size of each encoded element (and of the entire database) increases by a modest multiplicative factor. For databases with number of elements between 2^{18} and 2^{24}, this factor conservatively lies between 2 and 5. There is a similar blow-up in offline download and query response sizes.

We note that the increase in the LWE modulus requires an increase in the LWE dimension. Using the lattice estimator of Albrecht, Player, and Scott [2], we choose $n = 2520$ to ensure 128 bit LWE security. We provide examples of parameters with which to instantiate our schemes in Section 7.

Simple and practical SPIR. We provide a more unified approach to SPIR than previous work, achieving database privacy for PIR schemes via simple modifications which do not involve the

[4] The standard notion requires that the client's view should be statistically indistinguishable when interacting with any two databases DB and DB', as long as these databases coincide on the client's query positions.

use of other cryptographic primitives (e.g., [23, 48]). The resulting SPIR schemes are very easy to implement on top of the underlying PIR schemes while remaining practically feasible to run. Furthermore, they do not require additional hardness assumptions, except for the adoption of the random oracle model.

Limitations of our approach relative to database format. In the original SimplePIR [32] scheme, the database $\mathbf{D} \in \mathbb{Z}_p^{\sqrt{m} \times \sqrt{m}}$ is arranged as a square, so each row consists of $\sqrt{m}$ elements. When retrieving a record in SimplePIR, a client is in fact able to retrieve an *entire row*, and may then discard all elements except for the desired one. In the SPIR setting this is of course unacceptable, which is why we require the condition that each row must contain a single element, as initially proposed in FrodoPIR [20] where the database is parsed as $\mathbf{D} \in \mathbb{Z}_p^{m \times 1}$ (or, more generally, as $\mathbf{D} \in \mathbb{Z}_p^{m \times \omega}$, with ω defining the size of each record). The downside is a communication imbalance (particularly when $\omega \ll m$), since the query size becomes $\Theta(m)$, rather than $\Theta(\sqrt{m})$, where m is the number of records in the database. However, in FrodoPIR the offline hint to be downloaded by the client, and the query response, both have size independent of m. This leads to lower financial costs in commercial (e.g., AWS) deployments, since uploads, which dominate online communication, are free, and downloads are expensive.

Compatibility with recursive and hintless variants of PIR. While SimplePIR leads to the issue described above, its recursive variant, known as DoublePIR [32], allows us to avoid this problem, since clients only retrieve a single database element per query. It is not trivial to adapt DoublePIR to the SPIR setting using the same techniques, since DoublePIR queries consist of two levels of SimplePIR composed in a non-black-box way. However, we introduce a modified scheme and prove its database privacy in Section 6. We note that, for technical reasons related to database privacy, our adaptation of DoublePIR is suited to scenarios where the database has many elements (e.g., 2^{36} or more), which is nevertheless aligned with the motivation behind DoublePIR: while online costs are higher than in SimplePIR, and the offline hint is concretely large, the hint size is independent of the number of database elements, so download costs scale better for very large databases.

Other variants of LWE-based PIR which build on SimplePIR/FrodoPIR and DoublePIR have been proposed in which clients do not download any offline hint, often greatly reducing total client download at the cost of extra online communication and/or computation. Two main approaches have been proposed: the "bootstrapping" approach [31, 38], in which the server homomorphically computes the portion of the hint needed by the client at each query; and the "packing" approach [44, 42], in which the large LWE response ciphertexts are compressed into a smaller RLWE ciphertext via a LWE-to-RLWE packing algorithm. While we do not present the details of such constructions, we observe that the general modifications introduce no extra leakage, so database privacy can follow straightforwardly from that of the schemes *with* offline hints. In fact, the amount of leakage is often reduced, since the client may not receive the complete offline hint. However, adapting these schemes to the SPIR setting must be done with care, since some additional optimizations can be problematic. For instance, we note that the efficient NTT-based hint computation proposed in YPIR [44] involves the use of a *structured* matrix $\mathbf{A}$, in place of a randomly sampled one, which violates a critical condition for database privacy in our proofs.

Practical implementation. We implement our schemes built on top of FrodoPIR and provide an experimental performance analysis in Section 7. Our source code is available anonymously[5], allowing to compare the concrete communication and computation costs of FrodoPIR with our symetrically private counterparts. In particular, we obtain communication costs that are essentially a factor of 2 higher than the original scheme, and computation costs that increase roughly by a factor of 4. Considering that the underlying PIR scheme is considered to be practical for real-world use-cases, we view these performance magnifications as viable for usage in privacy-sensitive contexts.

1.2 Technical Overview

LWE-based PIR. In the family of PIR schemes introduced with FrodoPIR and SimplePIR, a database with m elements is represented as a matrix $\mathbf{D}_\rho^{m_{\text{row}} \times m_{\text{col}}}$, where $m_{\text{row}} \cdot m_{\text{col}} = m$ each element is encoded in an entry of size $\log \rho$. The original SimplePIR formulation set $m_{\text{row}} = m_{\text{col}} = \sqrt{m}$, while in FrodoPIR the choice was $m_{\text{row}} = m$ and $m_{\text{col}} = 1$. At a high level, a client's query $\mathbf{q}$ for an element in the i-th row consists of a vector of m_{row} LWE ciphertexts, with entry i encrypting 1 and all other entries encrypting 0. For the response, the server simply computes $\mathbf{q}^\top \cdot \mathbf{D}$ and sends the result to the client, with the linearly homomorphic property of the LWE encryption ensuring correct decryption.

As mentioned, the SimplePIR database format becomes problematic for the SPIR setting because the client retrieves an entire row of the database, in which only one of many elements was queried. For this reason, we adopt the FrodoPIR convention that the database holds one element per row (note that this means query size scales linearly in the size of the database). To deal with large elements, instead of increasing the plaintext modulus ρ, which results in a very large ciphertext modulus, one can simply decompose each w-bit element into $\omega = w/\log \rho$ blocks of size $\log \rho$ and pack ω entries into each row, so that the database becomes $\mathbf{D} \in \mathbb{Z}_\rho^{m \times \omega}$.

The main barrier to database privacy in such schemes is due to the fact that, upon decrypting the response, the client obtains the desired element *and* their LWE error term multiplied by the database matrix. This last term, of the form $\mathbf{e}^\top \cdot \mathbf{D}$, allows the client to obtain partial information about other database elements.

Noise flooding. As we have seen, query responses in LWE-based PIR schemes have the form of a vector $\mathbf{v}^\top + \mathbf{e}^\top \cdot \mathbf{D}$ over $\mathbb{Z}_q$, where $\mathbf{v}$ encodes the desired record (scaled up by some factor Δ), and $\mathbf{e}^\top \cdot \mathbf{D}$ is an error term that is small enough so that dividing the response by Δ and rounding to the nearest integer yields the desired record, i.e., the error term can be rounded off. The goal when aiming for database privacy is to hide the database-dependent error from the client.

In our case, the noise flooding technique consists of having the server sample a larger error $\mathbf{e}_{\mathsf{flood}}$, called the flooding noise, and add it to the response. The flooded response becomes $\mathbf{v}^\top + \mathbf{e}^\top \cdot \mathbf{D} + \mathbf{e}_{\mathsf{flood}}^\top$, and the idea is that $\mathbf{e}_{\mathsf{flood}}$ should make $\mathbf{e}^\top \cdot \mathbf{D}$ unnoticeable to the client, thus hiding the leakage of database contents. This is commonly formalized by requiring that $\mathbf{e}^\top \cdot \mathbf{D} + \mathbf{e}_{\mathsf{flood}}^\top$ and $\mathbf{e}_{\mathsf{flood}}^\top$ should be *statistically indistinguishable*. On the other hand, the larger $\mathbf{e}_{\mathsf{flood}}^\top$ is, the more the scaling factor Δ will have to increase to account for the noise, which worsens the scheme's parameters.

Replacing statistical distance with Rényi divergence. The key issue that arises when aiming for statistical indistinguishability between $\mathbf{e}^\top \cdot \mathbf{D} + \mathbf{e}_{\mathsf{flood}}^\top$ and $\mathbf{e}_{\mathsf{flood}}^\top$ is that the standard deviation σ of $\mathbf{e}_{\mathsf{flood}}$ must be superpolynomial in the security parameter to ensure negligible statistical distance.

[5] https://anonymous.4open.science/r/spir-1C75, https://anonymous.4open.science/r/spir_ro-6BFB

Correctness then requires that the ciphertext modulus must also be superpolynomial, to account for the error term, resulting in a large blow-up in memory and bandwidth requirements, as well as computations over larger values. Lastly, the superpolynomial LWE modulus-to-noise ratio affects security, which results in the need for larger LWE parameters to compensate.

We now illustrate why working with the Rényi divergence results in smaller parameter increases. Like the statistical distance, the Rényi divergence has a property that helps us bound the probability of a bad event happening in the security game. Roughly, this property states that for any event E and distributions P and Q, we have $P(E) \leq (Q(E) \cdot \exp(\delta))^c$, where δ is the Rényi divergence of P from Q and c is a constant close to 1. Thus, while the statistical distance must be negligible in the security parameter λ, even a Rényi divergence that *grows logarithmically* in λ is sufficient to preserve negligible probabilities. With a more careful analysis, this leads to security proofs that impose much less stringent requirements on the size of the flooding noise. For instance, the statistical distance between shifted discrete gaussian distributions with mean shift c and standard deviation σ scales with c/σ, while the Rényi divergence scales with $(c/\sigma)^2$. Thus, for this natural choice of flooding noise distribution, the former case requires the noise magnitude to be super-polynomial, while the latter does not (as long as c is polynomially bounded), which allows one to use much smaller LWE parameters.

An inherent technical difficulty is that the above probability preservation property does not work well with decision problems, where winning probabilities are large, e.g., close to $1/2$, but rather with search problems where winning probabilities should be negligible. However, it is naturally desirable to have security definitions based on *indistinguishability*, rather than *unpredictability*. To overcome this issue, we resort to the intermediate "one-way" database privacy notion mentioned above.

Search-to-decision reduction. As mentioned, the public sampleability framework [7, 9] does not seem appropriate for our work, as will become apparent. We therefore draw inspiration from the two step solution of [11] by first basing database privacy on the hardness of a search problem ("one-way" database privacy), and later showing how to transform a scheme satisfying this more modest security property into one that satisfies a stronger notion based on the hardness of a decision problem. The parameters that result from the first security proof will benefit from the improved Rényi divergence analysis, while the search-to-decision reduction consists of performing a simple and efficient encoding process on the database, based on a Fujisaki-Okamoto transformation [25].

Our encoding and decoding of database elements makes use of a random oracle hash function. We also propose the replacement of the random oracle with a strong extractor, which gives us database privacy under stronger assumptions on the database distribution (in fact, we need strong *average-case* extractors, but we omit this detail here for simplicity). A strong extractor E is a function that takes as input a random seed d and a source x, and whose output on sources with high min-entropy (inputs whose value is hard to guess) is statistically close to uniform. Informally, if x has high min-entropy then $(d, \mathsf{E}(d, x)) \approx (d, U)$, where U is uniform over the output space and $\approx$ denotes statistical indistinguishability.

The basic idea behind our transformation is the following: suppose that the value of any database entry $\mathbf{d}$ that hasn't been queried is *hard to guess*. Consider the new encoding of $\mathbf{d}$ as $(\mathbf{b}, \mathsf{H}(\mathbf{b}) + \mathbf{d})$, where $\mathbf{b}$ is a random value and H is a random oracle or an extractor. Then, the behavior of H and the guarantee that $\mathbf{b}$ is hard to guess ensures that $\mathbf{d}$ is appropriately masked by $\mathsf{H}(\mathbf{b})$, so that an adversary learns nothing about $\mathbf{d}$.

Dealing with hint leakage. In the original FrodoPIR and SimplePIR schemes, clients receive a hint in the offline phase which contains information about the database. Even in versions of these

schemes with no offline download [31, 38, 44, 42], there is still partial information about the hint which is received by the client later on. Therefore, we must take this leakage into account when arguing database privacy.

The hint in these schemes is a matrix $\mathbf{H} = \mathbf{A} \cdot \mathbf{D}$ with entries in $\mathbb{Z}_q$, where $\mathbf{A}$ is sampled (pseudo)randomly by the server and $\mathbf{D}$ encodes the database. Under certain conditions, a variant of the leftover hash lemma (which we give in Section 2) is sufficient to argue that replacing the hint with a database-independent random matrix would not significantly alter an adversary's advantage. In particular, it is sufficient to assume that the each column of $\mathbf{D}$ has sufficient min-entropy and that columns are independently distributed. Moreover, the encoding process described above, where the server adds fresh randomness to the database encoding, will help to ensure that these requirements hold.

2 Preliminaries

Notation. We assume all random variables are real-valued (in most cases they will even be discrete or finite). When no ambiguity arises, we confuse random variables with probability distributions. The probability mass function of a discrete random variable X is denoted P_X. We write $x \leftarrow P$ when x is sampled according to the probability distribution P and $x \xleftarrow{\$} A$ when x is sampled uniformly at random from the set A. The uniform distribution over a set A is denoted U_A. If $A = \{0,1\}^d$ we write U_d as long as no confusion arises. We implicitly assume all random variables are integrable, meaning that their expectation is well-defined and finite. The probability of an event E is denoted by $\Pr[E]$. We write $X \perp\!\!\!\perp Y$ to denote that X and Y are independently distributed.

We use $\mathcal{D}_{\mu,\sigma}$ (resp. $\mathcal{D}_{\mathbb{Z},\mu,\sigma}$) to denote the continuous (resp. discrete) gaussian distribution with parameters μ, σ. When $\mu = 0$ we may omit it from the notation.

We say an algorithm $\mathcal{A}$ is PPT if it runs in probabilistic polynomial time. We write negl to symbolize some negligible function. We use the notation $X \approx Y$ (resp. $X \approx_c Y$) when X and Y are statistically (resp. computationally) indistinguishable (note that we confuse random variables with ensembles, indexed by a security parameter, for simplicity).

We use log to refer to the base 2 logarithm, while the natural logarithm is written as ln. The symbol exp denotes the exponential function.

Vectors (resp. matrices) are generally denoted in lowercase (resp. uppercase) boldface letters. The i-th component of a vector $\mathbf{v}$ is denoted v_i. For a matrix $\mathbf{M}$ we denote its i-th row by $\mathbf{M}[i,\cdot]$, its j-th column by $\mathbf{M}[\cdot,j]$ and its entry in position (i,j) by $\mathbf{M}[i,j]$. The euclidean norm of a vector $\mathbf{v}$ is denoted by $\|v\|_2$, while the max-norm of $\mathbf{v}$, defined as $\max_i |v_i|$, is written as $\|\mathbf{v}\|_\infty$, where $|\cdot|$ denotes absolute value. We denote the concatenation of vectors $\mathbf{v}$ and $\mathbf{u}$ by $\mathbf{v}||\mathbf{u}$.

A set A's cardinality is denoted by $|A|$. The set $\{1,\ldots,n\}$ is denoted by $[n]$. The symbol $\propto$ denotes proportionality, i.e., $f \propto g$ when there exists a constant k such that $f(x) = k \cdot g(x)$ for all x. We may abuse notation and write $f(x) \propto g(x)$ instead of $f \propto g$ when no confusion arises.

We use the informal notation $a \ll b$ (resp. $a \gg b$) to indicate that a is much smaller (resp. larger) than b, and we write $a \lesssim b$ to denote that $a \leq b + \epsilon$ for some $\epsilon \ll b$.

2.1 Probability

Proposition 2.1 (Hoeffing bound)**.** *Let $X_1,\ldots,X_n$ be independent random variables such that $a_i \leq X_i \leq b_i$ almost surely. Let $S_n = \sum_{i=1}^n X_i$. Then, for all $t > 0$,*

$$\Pr\left[|S_n - \mathbb{E}[S_n]| \geq t\right] \leq 2\exp\left(-\frac{2t^2}{\sum_{i=1}^n (b_i - a_i)^2}\right).$$

We shall use the discrete gaussian as our flooding noise distribution.

Definition 2.2 (Discrete gaussian distribution)**.** *The* discrete gaussian distribution *over* $\mathbb{Z}$ *with parameters* μ, σ *is the distribution whose density satisfies*

$$\mathcal{D}_{\mathbb{Z},\mu,\sigma}(x) \propto \exp\left(\frac{(x-\mu)^2}{2\sigma^2}\right).$$

When $\mu = 0$ *we may simply write* $\mathcal{D}_{\mathbb{Z},\sigma}$*.*

The inequality below states well-known concentration bounds for continuous gaussians, which also applies to discrete gaussians (see e.g. [46, Lemma 2.8]).

Proposition 2.3 (Gaussian concentration bounds)**.** *If* X *is a gaussian or discrete gaussian random variable with mean* 0 *and variance* σ^2*, then*

$$\Pr\left[|X| \geq c\right] \leq \exp\left(-\frac{c^2}{2\sigma^2}\right).$$

Definition 2.4 (Statistical distance)**.** *The* statistical distance *between two random variables* X *and* Y *is given by*

$$\mathsf{SD}(X,Y) = \sup_T \left|\Pr[T(X)=1] - \Pr[T(Y)=1]\right|,$$

where the supremum is taken over all test functions T*. We may confuse random variables with probability distributions when no ambiguity arises.*

It is simple to prove that the statistical distance between discrete random variables X and Y over $\mathcal{X}$ satisfies the following equation:

$$\mathsf{SD}(X,Y) = \frac{1}{2}\sum_{a\in\mathcal{X}} \left|P_X(a) - P_Y(a)\right|.$$

The following basic property illustrates the usefulness of the statistical distance for security proofs of cryptographic protocols. Informally, it allows us to argue that if the distribution of the execution of a *real* protocol and an *ideal* one are statistically close, then the probability that "something bad happens" (e.g., an attacker recovers some secret information) is similar in both situations.

Proposition 2.5 (Probability preservation)**.** *For any measurable event* E *and distributions* P *and* Q *we have* $P(E) \leq Q(E) + \mathsf{SD}(P,Q)$*.*

The following property provides an easy way of obtaining upper bounds on the statistical distance of random variable tuples.

Proposition 2.6. *Let* (X_i, Y_i) *be discrete random variables in* $\mathcal{X}\times\mathcal{Y}$*,* $i = 1, 2$*, such that* $\mathrm{Supp}(X_1) \subseteq \mathrm{Supp}(X_2)$*. Then*

$$\mathsf{SD}((X_1,Y_1),(X_2,Y_2)) \leq \mathsf{SD}(X_1,X_2) + \mathop{\mathbb{E}}_{x\sim X_1}\left[\mathsf{SD}(Y_1|X_1=x, Y_2|X_2=x)\right].$$

To contrast our Rényi divergence analysis with the usual method of proving security using the statistical distance, it is useful to recall the some bounds on the statistical distance between discrete gaussians.

Proposition 2.7 (Statistical distance between discrete gaussians)**.** *The statistical distance between discrete gaussian distributions with shift $c \in \mathbb{Z} \setminus \{0\}$ satisfies*

$$\min\left\{1, \frac{1}{6\sigma}\right\} \leq \mathsf{SD}(\mathcal{D}_{\mathbb{Z},c,\sigma}, \mathcal{D}_{\mathbb{Z},\sigma}) \leq \frac{|c|}{2\sigma}.$$

We now turn to defining the Rényi divergence and listing some of its properties.

Definition 2.8 (Rényi divergence)**.** *Let P, Q be two discrete probability distributions such that* $\mathrm{Supp}(P) \subseteq \mathrm{Supp}(Q)$*, and let $a \neq 1$ be finite and positive. The* Rényi divergence of order *a of P from Q is given by*

$$\mathrm{RD}_a(P||Q) := \frac{1}{a-1} \ln\left(\sum_{x \in \mathrm{Supp}(P)} \frac{P(x)^a}{Q(x)^{a-1}}\right).$$

The definition above is generalized in the expected way to continuous distributions by replacing probabilities with densities and sums with integrals.

Below we recall the well-known data processing inequality. A proof for the Rényi divergence can be found in [22, Theorem 1].

Proposition 2.9 (Data processing inequality)**.** *Let $\mathcal{F}(P, Q)$ stand for the Rényi divergence of any order $a \in (1, \infty)$ or for the statistical distance between P and Q. Let P_X, Q_X be two distributions for a random variable X, and let P_Y, Q_Y be the corresponding distributions of the random variable $Y = f(X)$ induced by the (possibly randomized) function f. Then, $\mathcal{F}(P_Y, Q_Y) \leq \mathcal{F}(P_X, Q_X)$.*

Upper bounds on the Rényi divergence between two continuous gaussians [27] and two discrete gaussians [12], are given below.

Proposition 2.10 (Rényi divergence for discrete gaussians)**.** *For $\sigma > 0, c \in \mathbb{R}$ and $a \in [1, \infty)$ we have* $\mathrm{RD}_a(\mathcal{D}_{\mathbb{Z},c,\sigma}||\mathcal{D}_{\mathbb{Z},\sigma}) \leq \frac{ac^2}{2\sigma^2}$*, with equality if $c \in \mathbb{Z}$.*

Proposition 2.11 (Rényi divergence for product distributions [22, Thm. 28])**.** *For (P_n, Q_n) pairs of distributions, $N \in \mathbb{N}$ and $a \in [0, \infty]$:*

$$\sum_{n=1}^{N} \mathrm{RD}_a(P_n||Q_n) = \mathrm{RD}_a(P_1 \times ... \times P_N||Q_1 \times ... \times Q_N).$$

Proposition 2.12 (Right convexity [22, Thm. 12])**.** *For any $a \in [0, \infty]$, for any probability distributions P, Q_0, and Q_1, and any $\lambda \in (0, 1)$ we have*

$$\mathrm{RD}_a(P||(1-\lambda)Q_0 + \lambda Q_1) \leq (1-\lambda)\mathrm{RD}_a(P||Q_0) + \lambda \mathrm{RD}_a(P||Q_1).$$

Proposition 2.13 (Joint quasi-convexity [22, Thm. 13])**.** *For any $a \in [0, \infty]$, for any two pairs of distributions (P_0, Q_0) and (P_1, Q_1), and any $\lambda \in (0, 1)$ we have*

$$\mathrm{RD}_a((1-\lambda)P_0 + \lambda P_1||(1-\lambda)Q_0 + \lambda Q_1) \leq \max\{\mathrm{RD}_a(P_0||Q_0), \mathrm{RD}_a(P_1||Q_1)\}.$$

We contrast the following with Proposition 2.5. Note that when using the statistical distance, we require that it be *negligible* in order to preserve negligible probabilities, while the Rényi divergence need only be at most *logarithmic* in the security parameter. However, the statistical distance property works well with events of higher probability (e.g. close to $\frac{1}{2}$) while the property below does not. Thus, it is harder to take advantage of when dealing with security notions based on distinguishing problems, where an adversary's probability of winning is close to $\frac{1}{2}$.

Proposition 2.14 (Probability preservation [22]). *Let $E \subseteq Supp(Q)$ be an event. For any $a \in (1,\infty)$ we have $P(E)^{\frac{a}{a-1}} \leq Q(E) \cdot \exp(\mathrm{RD}_a(P||Q))$.*

Proposition 2.15 (Multiplicativity [52, Prop. 2]). *Let P, Q be two distributions of a pair (Y_1, Y_2). Let P_1 (resp. Q_1) denote the marginal distribution of Y_1 under P (resp. Q). Let $P_{2|1}(\cdot|y_1)$ (resp. $Q_{2|1}(\cdot|y_1)$) denote the conditional distribution of Y_2 given that $Y_1 = y_1$ under P (resp. Q). Then, for $a \in (1,\infty)$,*

$$\exp(\mathrm{RD}_a(P||Q)) \leq \exp(\mathrm{RD}_a(P_1||Q_1)) \cdot \max_{y_1 \in \mathrm{Supp}(Y_1)} \exp(\mathrm{RD}_a(P_{2|1}(\cdot|y_1)||Q_{2|1}(\cdot|y_1))).$$

Intuitively, the min-entropy of a random variable quantifies its unpredictability, i.e., how hard it is to guess its value.

Definition 2.16 (Min-entropy). *Let X be a discrete random variable. The* min-entropy *of X, denoted by $H_\infty(X)$, is given by $H_\infty(X) := -\log p_{\max}(X)$, where $p_{\max}(X) := \max_x P_X(x)$. If $H_\infty(X) \geq k$, we say that X is a k-*source.

The average conditional min-entropy of X given Y quantifies the unpredictability X given some side information Y. From a cryptographic perspective, one may think of Y as some information that is leaked to an adversary in some protocol and that may influence their chances of success in guessing the value of X.

Definition 2.17 (Average conditional min-entropy). *Let (X, Y) be a pair of random variables. The* average conditional min-entropy *of X given Y, denoted by $\tilde{H}_\infty(X|Y)$, is given by $\tilde{H}_\infty(X|Y) = -\log \mathop{\mathbb{E}}_{y \sim Y} [p_{\max}(X|Y=y)]$.*

Definition 2.18 (Strong extractors). *A function $E : \mathcal{X} \times \{0,1\}^d \to \mathcal{Z}$ is a (k,ϵ)-*strong extractor *if for every k-source X supported on $\mathcal{X}$, it holds that*

$$\mathsf{SD}\left((U_d, E(X, U_d)), (U_d, U_{\mathcal{Z}})\right) \leq \epsilon.$$

A natural extension of the previous definition arises to model the presence of (probabilistic) side information about the source. For average case strong extractors, we require the previous condition to hold even when a distinguisher is given some information W which is potentially correlated with X.

Definition 2.19 (Average-case strong extractors). *$E : \mathcal{X} \times \{0,1\}^d \to \mathcal{Z}$ is an* average-case (k,ϵ)-strong extractor *if for every pair (X, W) supported on $\mathcal{X} \times \mathcal{W}$ such that $\tilde{H}_\infty(X|W) \geq k$, it holds that $\mathsf{SD}\left((U_d, E(X, U_d), W), (U_d, U_{\mathcal{Z}}, W)\right) \leq \epsilon$.*

2.2 Generalized Leftover Hash Lemma

In the LWE-based PIR schemes we consider, clients also access database leakage by receiving an offline hint $\mathbf{A} \cdot \mathbf{D} \in \mathbb{Z}_q$, with $\mathbf{A}$ being public and randomly generated and $\mathbf{D}$ encoding the database. Since reasoning about this leakage is critical for privacy, we provide a generalization of the well-known Leftover Hash Lemma (LHL) to families of functions which are not necessarily universal hash families. This weakening of requirements allows us to make LHL-style arguments for linear functions over $\mathbb{Z}_q$ for composite q (namely, with q as a power-of-two). While existing algebraic LHLs (e.g. [40, 45, 10]) are sufficient for our purposes, our statement is quite simple and is abstracted from the algebraic structure (we deal with arbitrary families of functions over arbitrary discrete sets, rather than linear functions over rings).

Lemma 2.20 (Generalized LHL with side information)**.** *Let $\mathcal{H}$ be a set of functions $\{h_i : \mathcal{X} \to \mathcal{Z}\}_{i\in[D]}$, with $|\mathcal{Z}| = M$. Let X, W be discrete random variables over $\mathcal{X}, \mathcal{W}$ respectively. Define*

$$\mathsf{Bad} := \left\{ (x_1, x_2) \in \mathcal{X}^2 \,\middle|\, \Pr_{h \xleftarrow{\$} \mathcal{H}} [h(x_1) = h(x_2)] > \frac{1}{M} \right\}$$

and $\mathsf{Good} := \mathcal{X}^2 \setminus \mathsf{Bad}$. Let $p^w_{\mathsf{Bad}} := \Pr_{x_1,x_2 \leftarrow X} [(x_1, x_2) \in \mathsf{Bad} | W = w]$. Then, for a uniformly random choice of $H \in \mathcal{H}$, we have

$$\mathsf{SD}((H, H(X), W), (H, U_{\mathcal{Z}}, W)) \le \frac{1}{2}\sqrt{M \mathop{\mathbb{E}}_{w\sim W} \left[p^w_{\mathsf{Bad}}\right]}.$$

Proof. We follow the outline of the standard LHL proof in Vadhan's monograph [56]. Define $CP(Z)$ to be the collision probability of Z, that is, the probability that two independent samples from Z coincide. If π denotes the probability mass function of Z, we have

$$CP(Z) = \Pr_{Z\sim Z'}[Z = Z'] = \sum_z \pi(z)^2 = \|\pi\|_2^2.$$

We also have

$$\begin{aligned}
CP((H, H(X))) &= CP(H) \cdot \Pr_{\substack{H \xleftarrow{\$} \mathcal{H} \\ x_1,x_2 \xleftarrow{\$} \mathcal{X}}} [H(x_1) = H(x_2)] \\
&= \frac{1}{D} \Pr[(x_1, x_2) \in \mathsf{Bad}] \cdot \Pr[H(x_1) = H(x_2) | (x_1, x_2) \in \mathsf{Bad}] \\
&+ \frac{1}{D} \Pr[(x_1, x_2) \in \mathsf{Good}] \cdot \Pr[H(x_1) = H(x_2) | (x_1, x_2) \in \mathsf{Good}] \\
&\le \frac{1}{D} \left(p_{\mathsf{Bad}} + \frac{1}{M} \right). \qquad (1)
\end{aligned}$$

Let v be the difference between the probability mass functions of $(H, H(X))$ and $(H, U_{\mathcal{Z}})$. Since $\mathsf{SD}((H, H(X)), (H, U_{\mathcal{Z}})) = \frac{1}{2}\|v\|_1$, our goal will be to upper bound $\|v\|_1$. We do this by first upper bounding $\|v\|_2$ and then invoking Cauchy-Schwarz. Note that

$$\begin{aligned}
\|v\|_2^2 &= \sum_{h,h(x)} \left(\Pr[(H, H(X)) = (h, h(x))] - \Pr[(H, U_{\mathcal{Z}}) = (h, h(x))] \right)^2 \\
&= \sum_{h,h(x)} \left(\Pr[(H, H(X)) = (h, h(x))]^2 - \frac{2}{MD} \Pr[(H, H(X)) = (h, h(x))] + \frac{1}{(MD)^2} \right) \\
&= CP((H, H(X))) - \frac{2}{MD} + \frac{1}{MD} \\
&\le \frac{p_{\mathsf{Bad}}}{D}, \qquad (2)
\end{aligned}$$

where the inequality follows from Equation (1). Let u be such that $u_i = \mathrm{sign}(v_i)$, where $\mathrm{sign}(x) = 1$ if $x \ge 0$ and -1 otherwise. Then,

$$\begin{aligned}
\|v\|_1 &= \sum_i |v_i| = \langle v, u \rangle \\
&\le \|v\|_2 \cdot \|u\|_2 \qquad \text{(by the Cauchy-Schwarz inequality)}
\end{aligned}$$

$$\leq \sqrt{\frac{p_{\mathsf{Bad}}}{D}} \cdot \sqrt{MD} \qquad \text{(by Equation (2))}$$
$$= \sqrt{Mp_{\mathsf{Bad}}}.$$

Recalling that $\mathsf{SD}((H, H(X)), (H, U_{\mathcal{Z}})) = \frac{1}{2}\|v\|_1$, we get that

$$\mathsf{SD}((H, H(X)), (H, U_{\mathcal{Z}})) \leq \frac{1}{2}\sqrt{Mp_{\mathsf{Bad}}}. \tag{3}$$

To conclude the proof we must condition on the side information W. Let $X_w = (X|W = w)$. Then,

$$\begin{aligned}
\mathsf{SD}((H, H(X), W), (H, U_{\mathcal{Z}}, W)) &= \mathop{\mathbb{E}}_{w\sim W}[\mathsf{SD}((H, H(X_w)), (H, U_{\mathcal{Z}}))] \\
&\leq \mathop{\mathbb{E}}_{w\sim W}\left[\frac{1}{2}\sqrt{Mp^w_{\mathsf{Bad}}}\right] && \text{(by Equation (3))} \\
&\leq \frac{1}{2}\sqrt{M \mathop{\mathbb{E}}_{w\sim W}[p^w_{\mathsf{Bad}}]} && \text{(by Jensen's inequality)}
\end{aligned}$$

as desired. □

2.3 Learning With Errors

The query privacy of the PIR schemes which underpin our SPIR schemes is based on the hardness of the decisional Learning with Errors (LWE) problem [51]. We therefore assume that the decisional LWE problem is hard for appropriate choices of error distribution, such as those selected in [20, 32].

Definition 2.21 (Decisional LWE)**.** *Let χ be a probability distribution and let $q, n, m > 0$ depend on λ. The* Decisional Learning with Errors problem*, denoted $\mathsf{LWE}_{q,n,m,\chi}$, is defined as follows. Given $(\mathbf{A}, \mathbf{t}) \in \mathbb{Z}_q^{n\times m} \times \mathbb{Z}_q^m$, decide whether*

$$\mathbf{t} \leftarrow \mathbf{s}^\top \cdot \mathbf{A} + \mathbf{e}^\top \quad \textit{or} \quad \mathbf{t} \leftarrow \mathbf{u}^\top, \tag{4}$$

where $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$, $\mathbf{s}^\top \leftarrow (\chi)^n$, $\mathbf{e}^\top \leftarrow (\chi)^m$, $\mathbf{u}^\top \xleftarrow{\$} \mathbb{Z}_q^m$.

2.4 SPIR

Following the general framework introduced in [50], we will base ourselves on stateful PIR schemes, where the PIR interactions are split into a query-independent offline phase and a query-dependent online phase. To fix some notation, we consider a database DB containing m elements, each consisting of w bits. Each entry $\mathsf{DB}[i]$ is associated with an index $i \in [m]$ that corresponds to its position in the database. Thus, viewing $\mathsf{DB}[i]$ as a vector of w bits, we denote by $\mathsf{DB}[i]_k$ the k-th bit of $\mathsf{DB}[i]$. We use Φ to denote a class of distributions over the database space. For $\phi \in \Phi$, ϕ_i is the marginal distribution of the element in position $i \in [m]$. The number of queries made by each client is assumed to be bounded a priori by a parameter $\ell \leq m$. For a set $I = \{i_1, \ldots, i_\ell\}$ and fixed values $\mathbf{d}_1, \ldots, \mathbf{d}_\ell$, we denote by $\phi_{i|I}$ the distribution of element i conditioned on $\mathsf{DB}[i_k] = \mathbf{d}_{i_k}, k \in [\ell]$.

Definition 2.22 (Stateful SPIR)**.** *A stateful SPIR protocol consists of an offline and an online phase, which are defined as follows.*

Offline Phase:

- $\mathsf{ssetup}(1^\lambda)$: *An algorithm run by the server that outputs some initialization parameters* ip.
- $\mathsf{cinit}(\mathsf{ip})$: *A client initialization algorithm run on parameters* ip. *Outputs a message* msg *to be sent to the server during the offline phase.*
- $\mathsf{spreproc}(\mathsf{ip}, \mathsf{DB}, msg)$: *A server preprocessing algorithm run on* ip, *the server database* DB, *and client message* msg. *Outputs a set of public parameters* pp *to be downloaded by the client.*
- $\mathsf{cpreproc}(\mathsf{ip}, \mathsf{pp})$: *A client preprocessing algorithm run on the server-generated public parameters* $(\mathsf{ip}, \mathsf{pp})$, *that outputs a state* st. *The state* st *always includes* $(\mathsf{ip}, \mathsf{pp})$ *and the randomness to be used by the client in all* ℓ *calls to the* query *algorithm (see below).*

Online Phase:

- $\mathsf{query}(\mathsf{st}, i)$: *A client algorithm that generates a query* q *for the item in the* i*-th index of the server database, and optionally returns an updated state* st'.
- $\mathsf{respond}(\mathsf{DB}, \mathsf{q})$: *A server algorithm that outputs a response* r *to be returned to the client.*
- $\mathsf{process}(\mathsf{st}, \mathsf{r})$: *A client algorithm that takes the server response* r, *and outputs a database element* x.

We have included the algorithm cinit for the sake of completeness. Stateful PIR schemes that omit the cinit algorithm are said to have client-independent preprocessing phases, as is the case in [20, 32]. From now on, we assume client-independent preprocessing. Moreover, we remark that the online algorithms are extended in a trivial natural way to handle multiple indices, queries, and responses at once.

Definition 2.23 (Correctness)**.** *Let* DB *be the server database, let* i *be an index that the client seeks to query during the online phase, and let* $\mathsf{DB}[i]$ *be the* i*-th entry of* DB. *We say a SPIR scheme is correct if the following inequality holds:*

$$\Pr\left[x = \mathsf{DB}[i] \;\middle|\; \begin{array}{l} \mathsf{ip} \leftarrow \mathsf{ssetup}(1^\lambda), \\ \mathsf{pp} \leftarrow \mathsf{spreproc}(\mathsf{ip}, \mathsf{DB}, \mathsf{cinit}(\mathsf{ip})), \\ \mathsf{st} \leftarrow \mathsf{cpreproc}(\mathsf{ip}, \mathsf{pp}), \\ \mathsf{q} \leftarrow \mathsf{query}(\mathsf{st}, i), \\ \mathsf{r} \leftarrow \mathsf{respond}(\mathsf{DB}, \mathsf{q}), \\ x \leftarrow \mathsf{process}(\mathsf{st}, \mathsf{r}) \end{array}\right] > 1 - \mathsf{negl}(\lambda).$$

Definition 2.24 (Query privacy)**.** *A SPIR protocol satisfies* $\mathit{IND\text{-}QP}$ *security with security parameter* λ *if, for all PPT adversaries* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, *it holds that*

$$\mathsf{Adv}^{\mathit{IND\text{-}QP}}(\mathcal{A}) := \left|\Pr[\mathsf{Expt}_{\mathcal{A}}^{\mathit{IND\text{-}QP}}(1^\lambda) = 1] - \frac{1}{2}\right| = \mathsf{negl}(\lambda)$$

where $\mathsf{Expt}_{\mathcal{A}}^{\mathit{IND\text{-}QP}}$ *is the experiment defined in Fig. 1.*

$\mathsf{Expt}^{\texttt{IND-QP}}_{\mathcal{A}}(1^\lambda)$

1: $\texttt{ip} \leftarrow \texttt{ssetup}(1^\lambda)$
2: $\texttt{pp} \leftarrow \texttt{spreproc}(\texttt{ip}, \mathsf{DB}, \texttt{cinit}(\texttt{ip}))$
3: $\texttt{st} \leftarrow \texttt{cpreproc}(\texttt{ip}, \texttt{pp})$
4: $i_0, i_1 \leftarrow \mathcal{A}_1(\mathsf{DB}, \texttt{ip}, \texttt{pp})$ // $\mathcal{A}$ *chooses 2 challenge indices*
5: $b \xleftarrow{\$} \{0,1\}$
6: $\texttt{q}_b \leftarrow \texttt{query}(\texttt{st}, i_b)$ // $\mathcal{A}$ *receives challenge query*
7: $b' \leftarrow \mathcal{A}_2(\mathsf{DB}, \texttt{ip}, \texttt{pp}, i_0, i_1, \texttt{q}_b)$ // $\mathcal{A}$ *guesses challenge bit*
8: **return** $b' = b$

Figure 1: Experiment for IND-QP

3 Database privacy

Prior work on symmetric PIR in the two-server setting outlines a general set of guarantees that are necessary to achieve to provide database privacy [39]. In the single-server setting, various works have shown that generic transformations from PIR to SPIR exist — with logarithmic numbers of queries [49], and with polynomial blow-up in the security parameter [21]. In more recent work, it has been noted that pairing PIR exchanges with an OPRF exchange results in a PIR scheme that achieves database privacy (where database elements are encrypted with keys that are learned during the OPRF protocol). However, direct security models for symmetric PIR are seldom described.

In order to give formal guarantees of security, we start by defining the concept of database privacy (and thus, symmetric PIR) in a game-based security notion. We define two privacy notions, one based on a search problem and the other based on a decision problem. Informally, we will first provide a scheme for which it's *hard to guess* unqueried elements for an adversary with no prior knowledge of the database (i.e. for a database with uniform prior distribution), and then we transform such schemes into ones for which an adversary *learns nothing new* about unqueried elements, regardless of prior knowledge. We consider both adversaries who select all their query indices at once (*selective* privacy), and adversaries who perform queries adaptively (*adaptive* privacy). On the game itself, we note that we must take into account the prior knowledge that an adversary may have about the data held by the server. Thus, we parameterize our security games with a prior database distribution.

The definitions of one-way database privacy given below are parameterized by a "guessing threshold" $s \in [w]$. The intuitive meaning of this privacy notion is that, if $\mathsf{DB}[i]$ has not been queried, then no adversary is able to guess the value of a prefix of $\mathsf{DB}[i]$ of length s, except with negligible probability in λ (note that, in particular, s must be super-logarithmic in λ – otherwise random guessing is sufficient to break security).

Definition 3.1 (Selective One-Way Database Privacy)**.** *A SPIR protocol satisfies selective* (ℓ, s, Φ)*-OW-DP security with security parameter* λ*, allowed prior distributions* Φ*, query bound* ℓ *and guessing threshold* s *if, for all* $\phi \in \Phi$ *and all PPT adversaries* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*, it holds that*

$$\mathsf{Adv}^{(\ell,s,\phi)\text{-sel-OW-DP}}(\mathcal{A}) := \Pr[\mathsf{Expt}^{(\ell,s,\phi)\text{-sel-OW-DP}}_{\mathcal{A}}(1^\lambda) = 1] = \mathsf{negl}(\lambda)$$

where $\mathsf{Expt}^{(\ell,s,\phi)\text{-sel-OW-DP}}_{\mathcal{A}}$ *is defined in Fig. 2.*

We now specify the stronger, adaptive version of the security game. When the flavor is not specified, we are referring to the adaptive case, as it is clearly the most desirable.

$\mathsf{Expt}_{\mathcal{A}}^{(\ell,s,\phi)\text{-sel-OW-DP}}(1^\lambda)/\mathsf{Expt}_{\mathcal{A}}^{(\ell,s,\phi)\text{-OW-DP}}$

1: $\mathsf{DB} \leftarrow \phi$
2: $\mathtt{ip} \leftarrow \mathtt{ssetup}(1^\lambda)$
3: $\mathtt{pp} \leftarrow \mathtt{spreproc}(\mathtt{ip}, \mathsf{DB}, \mathtt{cinit}(\mathtt{ip}))$
4: $\mathtt{st} \leftarrow \mathtt{cpreproc}(\mathtt{ip}, \mathtt{pp})$

5: $\{\mathsf{ind}_k\}_{k\in[\ell]} \leftarrow \mathcal{A}_1(\phi, \mathtt{st})$ // $\mathcal{A}$ chooses ℓ indices to query
6: $\{\mathtt{q}_k\}_{k\in[\ell]}, \mathtt{st} \leftarrow \mathtt{query}(\mathtt{st}, \{\mathsf{ind}_k\}_{k\in[\ell]})$
7: $\{\mathtt{r}_k\}_{k\in[\ell]} \leftarrow \mathtt{respond}(\mathsf{DB}, \{\mathtt{q}_k\}_{k\in[\ell]})$ // compute the ℓ query responses
8: $\{x_k\}_{k\in[\ell]} \leftarrow \mathtt{process}(\mathtt{st}, \{\mathtt{r}_k\}_{k\in[\ell]})$ // recover the ℓ database elements

9: **for** $k \in [\ell]$ **do**
10: $\quad \mathsf{ind}_k \leftarrow \mathcal{A}_1(\phi, \mathtt{st}, \{\mathtt{ind}_j, \mathtt{q}_j, \mathtt{r}_j, \mathtt{x}_j\}_{j\in[k-1]})$ // $\mathcal{A}$ chooses next index
11: $\quad \mathtt{q}_k, \mathtt{st} \leftarrow \mathtt{query}(\mathtt{st}, \mathsf{ind}_k)$
12: $\quad \mathtt{r}_k \leftarrow \mathtt{respond}(\mathsf{DB}, \mathtt{q}_k)$
13: $\quad x_k \leftarrow \mathtt{process}(\mathtt{st}, \mathtt{r}_k)$
14: **end for**

15: $(i, \mathsf{guess}) \leftarrow \mathcal{A}_2(\phi, \mathtt{st}, \{\mathtt{ind}_k, \mathtt{q}_k, \mathtt{r}_k, \mathtt{x}_k\}_{k\in[\ell]})$ // $\mathcal{A}$ outputs a guess
16: **return** $(\mathsf{DB}[i]_1, \ldots, \mathsf{DB}[i]_s) = \mathsf{guess} \wedge i \notin \{\mathsf{ind}_k\}_{k\in[\ell]}$

Figure 2: Experiment for selective OW-DP, highlighted in yellow, for adaptive OW-DP, highlighted in blue. Non-highlghted parts are common to both experiments.

Definition 3.2 (Adaptive One-Way Database Privacy)**.** *A SPIR protocol satisfies adaptive* (ℓ, s, Φ)*-OW-DP security with security parameter* λ*, allowed prior distributions* Φ*, query bound* ℓ *and guessing threshold* s *if, for all* $\phi \in \Phi$ *and all PPT adversaries* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$*, it holds that*

$$\mathsf{Adv}^{(\ell,s,\phi)\text{-OW-DP}}(\mathcal{A}) := \Pr[\mathsf{Expt}_{\mathcal{A}}^{(\ell,s,\phi)\text{-OW-DP}}(1^\lambda) = 1] = \mathsf{negl}(\lambda)$$

where $\mathsf{Expt}_{\mathcal{A}}^{(\ell,s,\phi)\text{-OW-DP}}$ *is defined in Fig. 2.*

We note that it may be possible for an adversary to learn more about unqueried elements by simply retrieving other correlated elements. The definitions below then ensure that nothing new is learned *except for* what can be inferred through such dependencies, as this is unavoidable.

Definition 3.3 (Selective Indistinguishability Database Privacy)**.** *A SPIR protocol satisfies selective* (ℓ, Φ)*-IND-DP security with security parameter* λ*, allowed prior distributions* Φ *and query bound* ℓ *if, for all* $\phi \in \Phi$ *and all PPT adversaries* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$*, it holds that*

$$\mathsf{Adv}^{(\ell,\phi)\text{-sel-IND-DP}}(\mathcal{A}) := \left|\Pr[\mathsf{Expt}_{\mathcal{A}}^{(\ell,\phi)\text{-sel-IND-DP}}(1^\lambda) = 1] - \frac{1}{2}\right| = \mathsf{negl}(\lambda)$$

where $\mathsf{Expt}_{\mathcal{A}}^{(\ell,\phi)\text{-sel-IND-DP}}$ *is defined in Fig. 3.*

$\mathsf{Expt}_{\mathcal{A}}^{(\ell,\phi)\text{-sel-IND-DP}}(1^\lambda)/\mathsf{Expt}_{\mathcal{A}}^{(\ell,\phi)\text{-IND-DP}}(1^\lambda)$

$\mathsf{DB} \leftarrow \phi$
$\mathtt{ip} \leftarrow \mathtt{ssetup}(1^\lambda)$
$\mathtt{pp} \leftarrow \mathtt{spreproc}(\mathtt{ip}, \mathsf{DB}, \mathtt{cinit}(\mathtt{ip}))$
$\mathtt{st} \leftarrow \mathtt{cpreproc}(\mathtt{ip}, \mathtt{pp})$

$\{\mathsf{ind}_k\}_{k\in[\ell]} \leftarrow \mathcal{A}_1(\phi, \mathtt{st})$ // $\mathcal{A}$ chooses ℓ indices to query
$\{\mathsf{q}_k\}_{k\in[\ell]}, \mathtt{st} \leftarrow \mathtt{query}(\mathtt{st}, \{\mathsf{ind}_k\}_{k\in[\ell]})$
$\{\mathsf{r}_k\}_{k\in[\ell]} \leftarrow \mathtt{respond}(\mathsf{DB}, \{\mathsf{q}_k\}_{k\in[\ell]})$ // compute the ℓ query responses
$\{x_k\}_{k\in[\ell]} \leftarrow \mathtt{process}(\mathtt{st}, \{\mathsf{r}_k\}_{k\in[\ell]})$ // recover the ℓ database elements

for $k \in [\ell]$ **do**
  $\mathsf{ind}_k \leftarrow \mathcal{A}_1(\phi, \mathtt{st}, \{\mathtt{ind}_j, \mathsf{q}_j, \mathsf{r}_j, \mathsf{x}_j\}_{j\in[k-1]})$ // $\mathcal{A}$ chooses next index
  $\mathsf{q}_k, \mathtt{st} \leftarrow \mathtt{query}(\mathtt{st}, \mathsf{ind}_k)$
  $\mathsf{r}_k \leftarrow \mathtt{respond}(\mathsf{DB}, \mathsf{q}_k)$
  $x_k \leftarrow \mathtt{process}(\mathtt{st}, \mathsf{r}_k)$
end for

$i \leftarrow \mathcal{A}_2(\phi, \mathtt{st}, \{\mathtt{ind}_k, \mathsf{q}_k, \mathsf{r}_k, \mathsf{x}_k\}_{k\in[\ell]})$ // $\mathcal{A}$ chooses challenge index
$b \xleftarrow{\$} \{0,1\}$
if $b = 0$ **then**
  $\mathsf{r} \leftarrow \mathsf{DB}[i]$ // select real database element
else
  $\mathsf{r} \leftarrow \phi_{i|\{\mathtt{ind}_k\}_{k\in[\ell]}}$ // compute fresh sample
end if
$b' \leftarrow \mathcal{A}_3(\phi, \mathtt{st}, \{\mathtt{ind}_k, \mathsf{q}_k, \mathsf{r}_k, \mathsf{x}_k\}_{k\in[\ell]}, \mathsf{r})$ // $\mathcal{A}$ outputs a guess
return $b' = b \wedge i \notin \{\mathtt{ind}_k\}_{k\in[\ell]}$

Figure 3: Experiment for selective IND-DP, highlighted in yellow, and for adaptive IND-DP, highlighted in blue. Non-highlighted parts are common to both experiments.

Definition 3.4 (Adaptive Indistinguishability Database Privacy)**.** *A SPIR protocol satisfies adaptive (ℓ, Φ)-IND-DP security with security parameter λ, allowed prior distributions Φ and query bound ℓ if, for all $\phi \in \Phi$ and all PPT adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$, it holds that*

$$\mathsf{Adv}^{(\ell,\phi)\text{-IND-DP}}(\mathcal{A}) := \left|\Pr[\mathsf{Expt}_{\mathcal{A}}^{(\ell,\phi)\text{-IND-DP}}(1^\lambda) = 1] - \frac{1}{2}\right| = \mathsf{negl}(\lambda)$$

where $\mathsf{Expt}_{\mathcal{A}}^{(\ell,\phi)\text{-IND-DP}}$ is defined in Fig. 3.

Remark 3.5. *We note here the reason why our security games do not seem to fit the public sampleability framework of [7, 9]. Roughly, public sampleability requires the existence of a public sampling algorithm which, on input the adversary's view in an execution of the security game and any bit b, generates fresh challenge samples from the distribution of that same execution of the security game with challenge bit b. In other words, the adversary is able to generate fresh samples for either challenge bit (for instance, in semantic security of PKE the adversary is able to generate fresh*

encryptions of either challenge message). In our case, however, the ability to generate challenge samples for the challenge bit $b = 0$ implies the triviality of our security game.

3.1 Reduction from Adaptive to Selective Database Privacy

We now state a generic reduction from the adaptive to the non-adaptive flavors for both security definitions. We remark on the important detail that the non-adaptive adversary we construct to witness the reduction has the additional property that it chooses its query indices *independently* of the public parameters in the scheme. This will prove useful in later results, as our schemes have public parameters which leak some information about the database.

Theorem 3.6 (Reduction from adaptive to selective privacy)**.** *Fix a SPIR protocol. Let $\mathcal{A}$ be a PPT adversary against the (ℓ, s, ϕ)-`OW-DP` (resp. (ℓ, ϕ)-`IND-DP`) security game with advantage $\mathsf{adv}_{\mathcal{A}}$. Then, there exists a PPT adversary $\mathcal{A}'$ against the (ℓ, s, ϕ)-`sel-OW-DP` (resp. (ℓ, ϕ)-`sel-IND-DP`) security game with advantage $\mathsf{adv}_{\mathcal{A}'}$ such that $\mathsf{adv}_{\mathcal{A}'} \geq \mathsf{adv}_{\mathcal{A}} / \binom{m}{\ell}$. In particular, there exists such an $\mathcal{A}'$ whose query indices are independent of the public parameters.*

Proof. Consider $\mathcal{A}'$ who first samples a random subset I of ℓ query indices from $[m]$, then receives the public parameters, retrieves the corresponding query responses and proceeds to simulates $\mathcal{A}$ (using the same public parameters) as follows: when $\mathcal{A}$ makes a query with index i, if $i \notin I$ then $\mathcal{A}'$ outputs a random guess if it's the one-way security game, else $\mathcal{A}$' outputs a random bit independently of the challenge; if $i \in I$, $\mathcal{A}'$ responds to the query made by $\mathcal{A}$ with the response it obtained in its own game for the query with index i; after all queries have been made and retrieved by $\mathcal{A}$, $\mathcal{A}'$ acts identically to $\mathcal{A}$.

Let p be the probability that all of $\mathcal{A}$'s indices are in I. Since I is a random choice out of $\binom{m}{\ell}$ possible subsets of $[m]$ with size ℓ, the probability of I matching any given subset of $[m]$ of size ℓ is $p = 1/\binom{m}{\ell}$. Clearly, in the one-way security game we have

$$\mathsf{adv}_{\mathcal{A}'} \geq p \cdot \mathsf{adv}_{\mathcal{A}}$$

and in the security game we have

$$\Pr\left[\mathcal{A}' \text{ wins}\right] = p \cdot \Pr\left[\mathcal{A} \text{ wins}\right] + \frac{(1-p)}{2} = p \cdot \left(\Pr\left[\mathcal{A} \text{ wins}\right] - \frac{1}{2}\right) + \frac{1}{2}.$$

This implies that

$$\mathsf{adv}_{\mathcal{A}'} \geq p \cdot \mathsf{adv}_{\mathcal{A}},$$

which concludes the proof. □

4 Our Schemes

We now present a SPIR protocol denoted SPIR, which satisfies one-way database privacy for databases with uniform prior distribution. It is directly obtained from FrodoPIR [20] by having the server add a flooding noise term to query responses, and is therefore equally simple. Since we follow the implementation of [20], we choose the same LWE error distribution. However, it is straightforward to generalize the scheme to other LWE error distributions. We note that query privacy follows easily by reducing the task of distinguishing queries to the task of solving decisional LWE [20, 32].

4.1 Setup

The database DB will be parsed as a matrix $\mathbf{D}$ with $\log \rho$ bits packed into each entry. $\mathsf{DB}[i]$ is thus encoded in the i-th row of $\mathbf{D}$. We assume that the client knows which index it would like to query. Moreover, we assume that there are C clients that will each launch a maximum of q_c queries against DB. Let n and q be the LWE dimension and modulus, respectively; let λ denote the security parameter. During preprocessing we make use of a pseudorandom generator $\mathsf{PRG}(\mu, n, m, q)$ that takes a seed $\mu \xleftarrow{\$} \{0,1\}^\lambda$ and outputs a matrix in $\mathbb{Z}_q^{n \times m}$. Let χ denote the uniform distribution over $\{-1, 0, 1\}$. Additions and multiplications are implicitly performed mod q, and $\lfloor x \rceil_\rho$ denotes $\lfloor x \cdot \frac{\rho}{q} \rceil$ mod ρ, where $\lfloor \cdot \rceil$ denotes rounding to the nearest integer with ties broken arbitrarily.

Server setup ($\mathsf{SPIR.ssetup}$)**.** The server constructs their database containing m elements, and samples a seed $\mu \in \{0,1\}^\lambda$.

Server preprocessing ($\mathsf{SPIR.spreproc}$)**.** In this phase, the server derives a matrix $\mathbf{A} \leftarrow \mathsf{PRG}(\mu, n, m, q)$, where $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$. It then runs $\mathbf{D} \leftarrow \mathsf{parse}(\mathsf{DB}, \rho)$, where parse encodes the DB into a matrix $\mathbf{D} \in \mathbb{Z}_\rho^{m \times \omega}$, and where $\omega = \lceil w / \log(\rho) \rceil$. The server stores $\mathbf{D}$. To generate public parameters, the server runs $\mathbf{M} \leftarrow \mathbf{A} \cdot \mathbf{D}$, and then publishes the pair $(\mu, \mathbf{M}) \in \{0,1\}^\lambda \times \mathbb{Z}_q^{n \times \omega}$ to a public repository accessible by clients.

Client preprocessing ($\mathsf{SPIR.cpreproc}$)**.** Each client downloads $(\mu, \mathbf{M})$ from the public repository, and runs $\mathbf{A} \leftarrow \mathsf{PRG}(\mu, n, m, q)$. The client then samples q_c vectors $\mathbf{s}_j \leftarrow (\chi)^n$ and $\mathbf{e}_j \leftarrow (\chi)^m$. Finally, it computes $\mathbf{b}_j^\top \leftarrow \mathbf{s}_j^\top \cdot \mathbf{A} + \mathbf{e}_j^\top \in \mathbb{Z}_q^m$ and $\mathbf{c}_j^\top \leftarrow \mathbf{s}_j^\top \cdot \mathbf{M} \in \mathbb{Z}_q^\omega$, for each $j \in [q_c]$, and stores the pairs internally as the set $X = \{(\mathbf{b}_j, \mathbf{c}_j)\}_{j \in [q_c]}$. The client's initial state is the set X together with the public parameters.

Client query generation ($\mathsf{SPIR.query}$)**.** For the index i that the client wishes to query, the client generates a vector $\mathbf{f}_i = (0, \ldots, 0, q/\rho, 0, \ldots, 0)$, i.e., the all-zero vector except where $\mathbf{f}_i[i] = q/\rho$. The client then pops a pair $(\mathbf{b}, \mathbf{c})$ from the internal state st that it maintains, and computes $\tilde{\mathbf{b}} \leftarrow \mathbf{b} + \mathbf{f}_i \in \mathbb{Z}_q^m$, and sends $\tilde{\mathbf{b}}$ to the server.

Server response ($\mathsf{SPIR.respond}$)**.** The server receives $\tilde{\mathbf{b}}$ from the client and samples $\mathbf{e}_{\text{flood}} \leftarrow (\mathcal{D}_{\mathbb{Z},\sigma})^\omega$. The server then responds with $\tilde{\mathbf{c}}^\top \leftarrow \tilde{\mathbf{b}}^\top \cdot \mathbf{D} + \mathbf{e}_{\text{flood}}^\top \in \mathbb{Z}_q^\omega$.

Client postprocessing ($\mathsf{SPIR.process}$)**.** The client receives $\tilde{\mathbf{c}}^\top$ and outputs $\mathbf{x} \leftarrow \lfloor \tilde{\mathbf{c}}^\top - \mathbf{c}^\top \rceil_\rho \in \mathbb{Z}_\rho^\omega$.

4.2 Correctness

It is easy to check that correctness holds whenever $\|\mathbf{e}^\top \cdot \mathbf{D} + \mathbf{e}_{\text{flood}}^\top\|_\infty < q/(2\rho)$. Since our implementation is built from that of FrodoPIR, we assume the components of $\mathbf{e}$ to be sampled from the uniform ternary distribution. We can then use Hoeffding bounds and gaussian concentration bounds, together with a union bound, to upper bound $\|\mathbf{e}^\top \cdot \mathbf{D}\|_\infty$ and $\|\mathbf{e}_{\text{flood}}^\top\|_\infty$, respectively. For instance, to ensure correctness with probability at least $1 - 2^{-32}$, and for database elements of size up to 16 KB, it is sufficient to ensure that $q > 16\rho^2 m + 16\rho\sigma$.

4.3 One-way Database Privacy

Before stating our privacy guarantee for SPIR, we state a corollary of Lemma 2.20 that will be sufficient to argue that the information leaked to the adversary about the database in the server preprocessing is negligible.

Corollary 4.1. *Let the modulus q be a power of two and assume the database is sampled uniformly at random. Let $\mathbf{D}$ denote the preprocessed database according to the* SPIR *scheme. Let $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ and $\mathbf{U} \in \mathbb{Z}_q^{n\times\omega}$ both be uniformly distributed, with $\mathbf{U}$ independent of $\mathbf{A}$ and $\mathbf{D}$. Then, for any choice of indices $i_1, \ldots, i_\ell$ (possibly depending on $\mathbf{A}$),*

$$\mathsf{SD}\left((\mathbf{A}, \mathbf{A}\cdot\mathbf{D}, \{\mathbf{D}[i_k, \cdot]\}_{k\in[\ell]}), (\mathbf{A}, \mathbf{U}, \{\mathbf{D}[i_k, \cdot]\}_{k\in[\ell]})\right) \leq \frac{\omega}{2}\sqrt{2^{n\log(q)-m+\ell}}.$$

Proof. We begin by noting that, even though the indices may be chosen adversarially after seeing $\mathbf{A}$, the corresponding database rows are independent of $\mathbf{A}$.

We view the set of matrices $\mathcal{A} \in \mathbb{Z}_q^{n\times m}$ as a family of hash functions with input space $\mathcal{X} = \mathbb{Z}_\rho^m$, where each matrix is seen as a linear function mapping the input (a column of $\mathbf{D}$) to $\mathbb{Z}_q^n$ in the natural way. We may then verify that for this case, the set Good (cf. Lemma 2.20) contains all pairs $(\mathbf{x}, \mathbf{x}') \in \mathcal{X}^2$ such that $\mathbf{x}$ and $\mathbf{x}'$ differ in at least one position by an odd number. This is due to the fact that if $(x_k - x'_k)$ is odd then it is coprime with q (and invertible mod q), and so $\mathbf{A}(\mathbf{x} - \mathbf{x}') = 0$ is equivalent to the condition that, for all $i \in [n]$,

$$\left(\mathbf{A}[i,k]\left(x_k - x'_k\right) = -\sum_{j\neq k}\mathbf{A}[i,j]\left(x_j - x'_j\right)\right)$$
$$\Leftrightarrow \left(\mathbf{A}[i,k] = -\left(x_k - x'_k\right)^{-1}\sum_{j\neq k}\mathbf{A}[i,j]\left(x_j - x'_j\right)\right)$$

Due to uniformity and independence of $\mathbf{A}$'s entries, this occurs with probability at most q^{-n}, which is the inverse of the size of the output space.

It then follows that for each fixed value $\mathbf{w}$ of $\mathbf{W} = \{\mathbf{D}[i_k, \cdot]\}_{k\in[\ell]}$, we have $p^{\mathbf{w}}_{\mathsf{Bad}} \leq 2^{-m+\ell}$ for each column of $\mathbf{D}$, since this is upper bounded by the probability of randomly selecting an element of $\mathbb{Z}_\rho^{m-\ell}$ with all even entries (recall that each $\mathbf{D}[\cdot, j]$ is assumed to be uniformly distributed with independent entries). By Lemma 2.20, it then follows that for all $j \in \omega$

$$\mathsf{SD}\left((\mathbf{A}, \mathbf{A}\cdot\mathbf{D}[\cdot, j], \{\mathbf{D}[i_k, \cdot]\}_{k\in[\ell]}), (\mathbf{A}, \mathbf{u}_j, \{\mathbf{D}[i_k, \cdot]\}_{k\in[\ell]})\right) \leq \frac{1}{2}\sqrt{2^{n\log(q)-m+\ell}}$$

where $\mathbf{u}_j$ is distributed uniformly and independently of $\mathbf{A}, \mathbf{D}[\cdot, j]$. Since the ω columns of $\mathbf{D}$ are assumed to be independent, it follows from Proposition 2.6 that

$$\mathsf{SD}\left((\mathbf{A}, \mathbf{A}\cdot\mathbf{D}, \{\mathbf{D}[i_k, \cdot]\}_{k\in[\ell]}), (\mathbf{A}, \mathbf{U}, \{\mathbf{D}[i_k, \cdot]\}_{k\in[\ell]})\right) \leq \frac{\omega}{2}\sqrt{2^{n\log(q)-m+\ell}}$$

as desired. □

The following theorem gives a one-way database privacy guarantee in the selective setting, and as a corollary gives security of SPIR against *adaptive* adversaries via Theorem 3.6 with an additional loss. However, we note that for this initial SPIR protocol, which will be improved upon in the next chapter, this loss is significant and, particularly in the adaptive case, prohibitive. This is because s is required to be large for security to hold, and if database entries are of moderate sizes then we obtain no guarantee that an adversary cannot guess an entire database row.

Remark 4.2. *From a technical standpoint, it is useful to assume that the adversary's query indices are chosen independently of the hint* $\mathbf{A} \cdot \mathbf{D}$ *when proving database privacy, in order to limit the control the adversary has over the leakage. By Theorem 3.6, it is enough to prove selective privacy against adversaries who choose query indices independently of the public parameters in order to obtain adaptive privacy.*

However, if we only want selective privacy, the loss from Theorem 3.6 is undesirable. Thus, we still allow adversaries to see the matrix $\mathbf{A}$ *before choosing their query indices in our selective privacy proofs. This enables the construction of schemes with selective privacy which avoid the aforementioned loss: clients can download the offline hint encrypted under a pseudorandom value (generated by a short seed) and, upon receiving their query responses, clients also receive the short seed to be able to decrypt. Thus, it is still possible for clients to perform the expensive offline download in advance, and for the server to only do the preprocessing work once and publish the encrypted result to a public repository, rather than sending the large offline hint to each client individually.*

Remark 4.3. *Although the server uses a PRG in our scheme to reduce communication, we will treat its output as truly random in our security proofs for simplicity (i.e., instead of receiving a PRG seed, the client receives a full random matrix* $\mathbf{A}$ *in the offline phase). It is easy to check that the use of a PRG leads to at most a negligible additive term in the adversary's advantage (if this were not the case, one could construct a PPT distinguisher to break the PRG's security property). We ignore this negligible term in our upper bounds on the adversary's advantage for simplicity, noting that the PRG seed can always be chosen large enough to ensure the desired level of security. Moreover, since our main constructions use random oracles, the PRG itself can be replaced by a random oracle.*

Theorem 4.4 ((ℓ, s, Φ)-`sel-OW-DP` security)**.** *Let* Φ *contain only the database distribution* ϕ *where all bits are i.i.d. uniform. Let* $\mathcal{A}$ *be an adversary against the* (ℓ, s, ϕ)-`sel-OW-DP` *security game for* SPIR *whose queries are chosen independently of* $\mathbf{A} \cdot \mathbf{D}$*. For any* $a \in (1, \infty)$*, it holds that*

$$\mathsf{Adv}^{(\ell,s,\phi)\text{-}\texttt{sel-OW-DP}}(\mathcal{A}) \leq \left(\exp\left(\frac{\ell\omega a(m\rho)^2}{2\sigma^2}\right) \cdot (m-\ell) \cdot \left(2^{-s}+\epsilon\right)\right)^{\frac{a-1}{a}},$$

where $\epsilon = \omega\sqrt{2^{n\log(q)-m+\ell+1}}$.

Proof. Consider the following security games:

Game 0. $\mathbf{G}^0$ is the (ℓ, s, ϕ)-`sel-OW-DP` security game from Fig. 2 for our SPIR scheme.

Game 1. $\mathbf{G}^1$ is the same game as $\mathbf{G}_0$ except for the following modification to how query responses are computed: upon receiving query $\tilde{\mathbf{b}}^\top = (\mathbf{s}^\top \cdot \mathbf{A} + \mathbf{e}^\top) + \mathbf{f}_i^\top$, sample $\mathbf{e}_{\text{flood}} \leftarrow (\mathcal{D}_{\mathbb{Z},\sigma})^\omega$ and respond with

$$\tilde{\mathbf{c}}^\top \leftarrow \mathbf{s}^\top \cdot \mathbf{A} \cdot \mathbf{D} + \mathbf{f}_i^\top \cdot \mathbf{D} + \mathbf{e}_{\text{flood}}^\top \in \mathbb{Z}_q^\omega.$$

Game 2. $\mathbf{G}^2$ is the same game as $\mathbf{G}^1$ except for the following modifications: the hint received by the adversary in the offline phase is $(\mathbf{A}, \mathbf{U})$, where $\mathbf{U}$ is uniform and independent of $\mathbf{A}$ and $\mathbf{D}$; upon receiving query $\tilde{\mathbf{b}}^\top = (\mathbf{s}^\top \cdot \mathbf{A} + \mathbf{e}^\top) + \mathbf{f}_i^\top$, sample $\mathbf{e}_{\text{flood}} \leftarrow (\mathcal{D}_{\mathbb{Z},\sigma})^\omega$ and respond with

$$\tilde{\mathbf{c}}^\top \leftarrow \mathbf{s}^\top \cdot \mathbf{U} + \mathbf{f}_i^\top \cdot \mathbf{D} + \mathbf{e}_{\text{flood}}^\top \in \mathbb{Z}_q^\omega.$$

Claim 4.5.

$$\mathsf{Adv}^{\mathbf{G}^0}(\mathcal{A}) \leq \left(\exp\left(\frac{\ell\omega a(m\rho)^2}{2\sigma^2}\right) \cdot \mathsf{Adv}^{\mathbf{G}^1}(\mathcal{A})\right)^{\frac{a-1}{a}}.$$

Proof. Let $\mathcal{T}_i = \left(\mathsf{DB}, \mathtt{ip}, \mathtt{pp}, \mathtt{st}, (\mathtt{ind}_j, \mathtt{q}_j, \mathtt{r}_j, \mathtt{x}_j)_{j\in[\ell]}, i, \mathtt{guess}, b\right)$ denote the trace containing the information generated in game $\mathbf{G}^i$, where b is the output bit. It is a consequence of Proposition 2.14 that

$$\mathsf{Adv}^{\mathbf{G}^0}(\mathcal{A}) \leq \left(\exp\left(\mathrm{RD}_a(\mathcal{T}_0||\mathcal{T}_1)\right) \cdot \mathsf{Adv}^{\mathbf{G}^1}(\mathcal{A})\right)^{\frac{a-1}{a}}.$$

Note that there exists a randomized algorithm that receives as input $\mathbf{D}$, the randomness used in $\mathcal{A}$'s queries, and the ℓ samples $(\mathbf{e}^\top \cdot \mathbf{D} + \mathbf{e}_{\text{flood}}^\top)$ or $(\mathbf{e}_{\text{flood}}^\top)$ and simulates, respectively, game $\mathbf{G}^0$ or $\mathbf{G}^1$ while generating the respective trace. Thus, by the data processing inequality, it suffices to bound the Rényi divergence between the distribution of these inputs to obtain a bound on the Rényi divergence between the traces. We can use the multiplicativity property of Proposition 2.15 to show that the Rényi divergence between the input distributions is at most

$$\max_{\substack{D\in\mathrm{Supp}(\mathbf{D})\\ e_1,\ldots,e_\ell\in\mathrm{Supp}(\mathbf{e})}} \mathrm{RD}_a\left((e_1^\top \cdot D + \mathbf{e}_{\text{flood}_1}^\top, \ldots, e_\ell^\top \cdot D + \mathbf{e}_{\text{flood}_\ell}^\top) \,\middle||\, (\mathbf{e}_{\text{flood}_1}^\top, \ldots, \mathbf{e}_{\text{flood}_\ell}^\top)\right).$$

By Proposition 2.11, this is at most

$$\ell \cdot \max_{\substack{D\in\mathrm{Supp}(\mathbf{D})\\ e\in\mathrm{Supp}(\mathbf{e})}} \mathrm{RD}_a\left((e^\top \cdot D + \mathbf{e}_{\text{flood}}^\top, \mathbf{e}_{\text{flood}}^\top)\right).$$

Noting that each of the ω components of $\mathbf{e}_{\text{flood}}$ has a discrete gaussian distribution, applying now Propositions 2.11 and 2.10 gives the bound

$$\mathrm{RD}_a(\mathcal{T}_0||\mathcal{T}_1) \leq \frac{\ell\omega a(m\rho)^2}{2\sigma^2},$$

and the proof of the claim is concluded. □

Claim 4.6.

$$\mathsf{Adv}^{\mathbf{G}^1}(\mathcal{A}) \leq (m-\ell)\cdot\left(2\epsilon + 2^{-s}\right),$$

where $\epsilon := \frac{\omega}{2}\sqrt{2^{n\log(q)-m+\ell+1}}$

Proof. Consider an adversary $\mathcal{A}'$ against $\mathbf{G}^1$ who receives the offline hint, feeds it to $\mathcal{A}$ and chooses the same queries as $\mathcal{A}$. Before receiving the query responses, $\mathcal{A}'$ also samples i' uniformly from the set of the remaining $m-\ell$ indices. $\mathcal{A}'$ then gives its query responses to $\mathcal{A}$, who proceeds to choose an index i and output a guess. If $i' \neq i$, $\mathcal{A}'$ outputs a fresh random guess, else it outputs the same guess as $\mathcal{A}$. Since $\mathcal{A}'$ guesses $i' = i$ with probability at least $\frac{1}{m-\ell}$, it is clear that

$$\mathsf{Adv}^{\mathbf{G}^1}(\mathcal{A}) \leq \mathsf{Adv}^{\mathbf{G}^1}(\mathcal{A}')\cdot(m-\ell). \tag{5}$$

It now suffices to upper bound the advantage of $\mathcal{A}'$, who chooses its query indices independently from $\mathbf{A}\cdot\mathbf{D}$ *and* chooses its guess index at random.

Consider now an adversary $\mathcal{B}$ who attempts to distinguish between $(\mathbf{A}, \mathbf{A}\cdot\mathbf{D})$ and $(\mathbf{A}, \mathbf{U})$ while obtaining some side information on $\mathbf{D}$ as described in Corollary 4.1 (by learning ℓ rows).

$\mathcal{B}$ first receives $\mathbf{A}$ and simulates $\mathcal{A}'$ to obtain the query indices $\{i_k\}_{k\in[\ell]}$ and the guess index i'. $\mathcal{B}$ then receives $\{\mathbf{D}[i_k,\cdot]\}_{k\in[\ell]}$ and $\mathbf{D}[i',\cdot]$. Then, for a randomly sampled bit b, $\mathcal{B}$ receives $\mathbf{A}\cdot\mathbf{D}$ if $b=1$ and $\mathbf{U}$ otherwise. $\mathcal{B}$ is then able to respond to the queries made by $\mathcal{A}'$ accordingly and to check if $\mathcal{A}'$ guessed $\mathbf{D}[i',\cdot]$ correctly or not. Indeed, if $b=1$ then $\mathcal{B}$ simulates $\mathcal{A}'$ in game $\mathbf{G}^1$, otherwise it simulates $\mathcal{A}'$ in game $\mathbf{G}^2$.

Let $p'_i = \mathsf{Adv}^{\mathbf{G}^i}(\mathcal{A}')$, $i=1,2$. If $\mathcal{A}'$ wins, then $\mathcal{B}$ outputs 1, otherwise $\mathcal{B}$ outputs a random bit. By Corollary 4.1, $\mathcal{B}$'s distinguishing advantage is at most ϵ. We have that

$$\begin{aligned}\epsilon &\geq \left|\Pr\left[\mathcal{B}\left(\mathbf{A},\mathbf{A}\cdot\mathbf{D}\right)=1\right]-\Pr\left[\mathcal{B}\left(\mathbf{A},\mathbf{U}\right)=1\right]\right| \\ &= \left|p'_1+\frac{1}{2}(1-p'_1)-p'_2-\frac{1}{2}(1-p'_2)\right| \\ &= \frac{1}{2}\left|p'_1-p'_2\right|.\end{aligned}$$

Thus, by Equation (5) we have $\mathsf{Adv}^{\mathbf{G}^1}(\mathcal{A}) \leq (m-\ell)\cdot\left(2\epsilon+\mathsf{Adv}^{\mathbf{G}^2}(\mathcal{A}')\right)$. Observe that in $\mathbf{G}^2$ the adversary learns no information about unqueried database entries. Thus, $\mathsf{Adv}^{\mathbf{G}^2}(\mathcal{A}') \leq \frac{1}{2^s}$, and the claim is proved. □

The result follows from the claims above. □

5 From Search to Decision-based Database Privacy

We now improve our database privacy guarantee by adapting the Fujisaki-Okamoto transformation [25] to go from a notion based on unpredictability to one based on indistinguishability. Our techniques were inspired by those of [11] in the context of Rényi noise flooding. The idea is simply to encode database elements with some redundant randomness, ensuring that as long as not *all* redundant randomness is recovered by the adversary, then no actual database information is leaked.

$\mathsf{SPIR}^{\mathsf{ro}}$. Henceforth, we let $\mathsf{F}:\{0,1\}^*\to\{0,1\}^w$ be a function modeled as a random oracle, accessible to both the server and the client. Consider the modification to our previous SPIR scheme described below.

$\mathsf{SPIR}^{\mathsf{ro}}$.ssetup. The server constructs their database DB containing m elements. For each entry $\mathsf{DB}[i]$, the server samples $\mathbf{b}_i \xleftarrow{\$} \{0,1\}^s$, where s is a function of λ. It then constructs the array $\widehat{\mathsf{DB}}$ such that

$$\widehat{\mathsf{DB}}[i] = \left(\mathbf{b}_i, \mathsf{F}\left(i,\mathbf{b}_i\right)\oplus\mathsf{DB}[i]\right)\in\{0,1\}^{s+w}$$

for every $i\in[m]$, and samples a seed $\mu\in\{0,1\}^\lambda$. Here, in the input to F, we confuse i with its representation in $\log m$ bits.

$\mathsf{SPIR}^{\mathsf{ro}}$.spreproc. In this phase, the server derives a matrix $\mathbf{A}\leftarrow\mathsf{PRG}(\mu,n,m,q)$, where $\mathbf{A}\in\mathbb{Z}_q^{n\times m}$. It then runs $\mathbf{D}\leftarrow\mathsf{parse}(\widehat{\mathsf{DB}},\rho)$, where parse encodes the $\widehat{\mathsf{DB}}$ into a matrix $\mathbf{D}\in\mathbb{Z}_\rho^{m\times\omega}$, and where $\omega=\lceil(w+s)/\log(\rho)\rceil$. The server stores $\mathbf{D}$.

To generate public parameters, the server runs $\mathbf{M}\leftarrow\mathbf{A}\cdot\mathbf{D}$, and then publishes the pair $(\mu,\mathbf{M})\in\{0,1\}^\lambda\times\mathbb{Z}_q^{n\times\omega}$ to a public repository accessible by clients.

$\mathsf{SPIR}^{\mathsf{ro}}$.process. The client receives $\tilde{\mathbf{c}}^\top$ and computes $\mathbf{x} \leftarrow \lfloor \tilde{\mathbf{c}}^\top - \mathbf{c}^\top \rceil_\rho \in \mathbb{Z}_\rho^\omega$. They then output $\mathbf{y} := \mathsf{F}(i, x_1, \ldots, x_s) \oplus (x_{s+1}, \ldots, x_{s+w}) \in \{0,1\}^w$, where x_k is the k-th bit of $\mathbf{x}$ and i is the queried index.

5.1 Database Privacy in the ROM

Below, we establish the indistinguishability-based database privacy of $\mathsf{SPIR}^{\mathsf{ro}}$, in the random oracle model.

Theorem 5.1 ((ℓ, Φ)-sel-IND-DP security)**.** *Let ϕ be any database distribution. Let $\mathcal{A}$ be an adversary against the (ℓ, ϕ)-sel-IND-DP security game for the $\mathsf{SPIR}^{\mathsf{ro}}$ scheme making at most q_{F} queries to F and whose queries are chosen independently of $\mathbf{A} \cdot \mathbf{D}$. For any $a \in (1, \infty)$, it holds that*

$$\mathsf{Adv}^{(\ell,\phi)\text{-sel-IND-DP}}(\mathcal{A}) \leq q_{\mathsf{F}} \cdot \left(\exp\left(\frac{\ell \omega' a (m\rho)^2}{2\sigma^2} \right) \cdot (m - \ell) \cdot \left(2^{-s} + \epsilon\right) \right)^{\frac{a-1}{a}},$$

where $\omega' = \lceil \frac{w+s}{\log(\rho)} \rceil$ and $\epsilon = \omega' \sqrt{2^{n \log(q) - m + \ell + 1}}$.

Proof. Consider the following games:

Game 0. $\mathbf{G}^0$ is the (ℓ, ϕ)-sel-IND-DP security game from Fig. 3 for our $\mathsf{SPIR}^{\mathsf{ro}}$ scheme

Game 1. $\mathbf{G}^1$ is the same game as $\mathbf{G}^0$ except for the following modification: If $\mathcal{A}$ queries F on input $(i, \mathbf{b}_i)$, for some unqueried index i, $\mathbf{G}^1$ aborts. Let the event that such a query occurs be denoted by abort.

Clearly,

$$\left| \Pr[\mathbf{G}^0_{\mathcal{A}} = 1] - \Pr[\mathbf{G}^1_{\mathcal{A}} = 1] \right| \leq \Pr[\mathsf{abort}]. \tag{6}$$

Moreover, given that F is a random oracle, then the adversary's view in $\mathbf{G}^1$ is independent of the chosen bit b (until abort occurs), since each unqueried $\mathsf{DB}[i]$ is completely masked by F's output. This means that

$$\Pr[\mathbf{G}^1_{\mathcal{A}} = 1] = \frac{1}{2}. \tag{7}$$

Equations (6) and (7) yield

$$\mathsf{Adv}^{(\ell,\phi)\text{-sel-IND-DP}}(\mathcal{A}) \leq \Pr[\mathsf{abort}]. \tag{8}$$

We now define an adversary $\mathcal{B}$ against the (ℓ, s, ϕ')-sel-OW-DP game for SPIR where ϕ' is the uniform distribution over $\{0,1\}^{m \times (s+w)}$.

Since $\mathcal{A}$'s query index choices are independent of $\mathbf{AD}$, there exists a (possibly randomized) function f that on input $\mathbf{A}$ returns $\mathcal{A}$'s query index choices. Note that $\mathcal{B}$ can simulate $\mathcal{A}$ with access to $\mathcal{A}$'s random tape. Thus, $\mathcal{B}$ computes $i_1, \ldots, i_\ell \leftarrow f(\mathbf{A})$ and queries its database on these positions to retrieve $\mathbf{d}_{i_1}, \ldots, \mathbf{d}_{i_\ell}$. Let $\mathbf{d}_{i_k}^{(s)}$ (resp. $\mathbf{d}_{i_k}^{(w)}$) denote the first s bits (resp. the last w bits) of $\mathbf{d}_{i_k}$, for $k \in [\ell]$.

$\mathcal{B}$ also samples $\mathbf{d}'_{i_1}, \ldots, \mathbf{d}'_{i_\ell}$ according to their joint distribution defined by ϕ. After this, $\mathcal{B}$ feeds $(\mathbf{A}, \mathbf{AD})$ to $\mathcal{A}$ who outputs the queries for indices $i_1, \ldots, i_\ell$, to which $\mathcal{B}$ replies with the responses it received for $\mathbf{d}_{i_1}, \ldots, \mathbf{d}_{i_\ell}$.

When $\mathcal{A}$ queries F on input $(i_k, \mathbf{d}_{i_k}^{(s)})$ inside the simulation, $\mathcal{B}$ responds to the query with $\mathbf{d}_{i_k}^{(w)} \oplus \mathbf{d}'_{i_k}$. (Essentially, $\mathcal{B}$ is programming F's output on that input so that the value of the database element retrieved by $\mathcal{A}$ is $\mathbf{d}'_{i_k}$, which is consistent with the distribution ϕ. Note also that the

responses to $\mathcal{A}$'s random oracle queries thus defined remain uniformly random and independent, as the $\mathbf{d}_{i_k}^{(w)}$ are i.i.d. uniform.) For the rest of $\mathcal{A}$'s queries to F, $\mathcal{B}$ responds with a fresh random value. Thus, $\mathcal{A}$'s view inside the simulation is identical to its view in the real game, as long as abort has not occurred.

If $\mathcal{A}$ queries F on a forbidden input, triggering abort, then $\mathcal{B}$ wins its security game with probability $1/q_{\mathsf{F}}$ by simply selecting a database index and one of $\mathcal{A}$'s inputs to F at random. Then,

$$\mathsf{Adv}^{(\ell,s,\phi)\text{-}\mathtt{sel\text{-}OW\text{-}DP}}(\mathcal{B}) \geq \Pr[\mathsf{abort}] \cdot \frac{1}{q_{\mathsf{F}}}$$

and taking Equation (8) together with Theorem 4.4 provides the desired bound. □

We reiterate that the result above implies database privacy for adaptive adversaries with an $\binom{m}{\ell}$ loss, as a consequence of Theorem 3.6.

Database privacy in the standard model. We note that the random oracle F in our $\mathsf{SPIR}^{\mathsf{ro}}$ scheme can be replaced by a strong average-case extractor E to obtain the same privacy guarantee. However, our security proof assumes a uniform prior database to deal with the hint leakage, since our LHL argument assumes the database has independent columns (this was true in the random oracle case, as long as forbidden inputs had not been queried). We leave it open to provide a security proof in the standard model for arbitrary prior database distributions.

We denote the modified scheme by $\mathsf{SPIR}^{\mathsf{sm}}$, and it is obtained from $\mathsf{SPIR}^{\mathsf{ro}}$ by simply replacing F with $\mathsf{E}_d : \{0,1\}^s \to \{0,1\}^w$, for some randomly sampled seed d which is published by the server as a public parameter. Thus, we now have

$$\widehat{\mathsf{DB}}[i] = (\mathbf{b}_i, \mathsf{E}_d(\mathbf{b}_i) \oplus \mathsf{DB}[i]) \in \{0,1\}^{s+w}.$$

Remark 5.2. *In cases where the database elements are large, and thus the extractor output must be large, an alternative is to use an extractor that produces a near-uniform short output, which can then be expanded using a PRG to a pseudorandom value that can mask an entire database record. This extract-then-expand approach allows us, in principle, to use a shorter extractor seed and source, thereby reducing storage and communication costs. For instance, practically efficient universal hash functions tend to have large seeds, which can be mitigated by this construction. For ease of exposition, we do not consider this alternative in our security proof.*

The theorem below gives the selective indistinguishability-based database privacy of $\mathsf{SPIR}^{\mathsf{sm}}$ in the standard model, under the assumption that the adversary has no prior knowledge about the database. We reiterate that adaptive security follows from Theorem 3.6.

Theorem 5.3 ((ℓ,Φ)-`sel-IND-DP` security)**.** *Let Φ contain only the database distribution ϕ where all bits are i.i.d. uniform. Suppose E is an average case (k,δ)-strong extractor. Let $\mathcal{A}$ be an adversary against the (ℓ,ϕ)-`sel-IND-DP` security game for the $\mathsf{SPIR}^{\mathsf{sm}}$ scheme whose queries are chosen independently of $\mathbf{A}\cdot\mathbf{D}$. For any $a \in (1,\infty)$ it holds that $\mathsf{Adv}^{(\ell,\phi)\text{-}\mathtt{sel\text{-}IND\text{-}DP}}(\mathcal{A}) \leq \delta$ as long as*

$$2^{-k} \geq \left(\exp\left(\frac{\ell\omega' a(m\rho)^2}{2\sigma^2}\right)\cdot(m-\ell)\cdot\left(2^{-s}+\epsilon\right)\right)^{\frac{a-1}{a}}, \tag{9}$$

where $\epsilon = \omega'\sqrt{2^{n\log(q)-m+\ell+1}}$ and $\omega' = \lceil\frac{w+s}{\log(\rho)}\rceil$.

Proof. Let i be the index chosen by $\mathcal{A}$ and let $\mathcal{V}$ denote $\mathcal{A}$'s view in the security game. The goal is then to show that, given $\mathcal{V}$, $\mathsf{E}_d(\mathbf{b}_i)$ looks sufficiently random to $\mathcal{A}$ so that $\mathsf{E}_d(\mathbf{b}_i) \oplus \mathsf{DB}[i]$ is seemingly uncorrelated with $\mathsf{DB}[i]$. We may even assume that $\mathsf{E}_d(\mathbf{b}_i) \oplus \mathsf{DB}[i]$ is included in $\mathcal{A}$'s view.

Note that the encoded database is distributed uniformly. By Equation (9) and Theorem 4.4, we know that $\tilde{H}_\infty(\mathbf{b}_i|\mathcal{V}) \geq k$. Since E is an average case (k, δ)-strong extractor, we have

$$\mathsf{SD}\left((d, \mathsf{E}_d(\mathbf{b}_i), \mathcal{V}), (d, \mathbf{u}, \mathcal{V})\right) \leq \delta,$$

where $\mathbf{u} \in \{0,1\}^w$ is uniformly random and independent of all else. By the data processing inequality,

$$\mathsf{SD}\left((d, \mathsf{DB}[i], \mathcal{V}), (d, \mathbf{u}, \mathcal{V})\right) \leq \delta,$$

where processing function corresponds to the component-wise $\oplus$ operation with $\mathsf{E}_d(\mathbf{b}_i) \oplus \mathsf{DB}[i]$ in the second component. □

6 Recursive SPIR Scheme

We now discuss how to extend our techniques to the recursive version of SimplePIR called DoublePIR, yielding a symmetrically-private scheme which we denote by DoubleSPIR.

Overview of DoublePIR. In DoublePIR [32], the hint size becomes independent of the number of elements m in the database, although it is concretely large, at the cost of some online overhead. This allows us to have a scheme where the client's offline download scales very well for very large databases. The idea behind DoublePIR is to note that, to retrieve a desired entry in position (i, j), one can execute an outer level of SimplePIR to obtain an encryption $\mathbf{a}$ of the i-th row, and then execute a second, inner level of SimplePIR over the inner database $\left[\mathbf{H}^\top \mid \mathbf{a}\right]$, where $\mathbf{H} = \mathbf{A} \cdot \mathbf{D}$ is the original SimplePIR hint, to obtain the j-th row of this database and recover the desired element (note that it is also necessary for the client to retrieve a hint about the inner database, which in particular depends on the outer answer vector).

To derive a symmetrically private version of DoublePIR, we note that the leakage comes from the error terms in both the outer and inner PIR queries, as well as from the inner hint. This inner hint depends on the database via the outer hint and outer answer vector.

Our idea. Given the discussion above, it is natural to deal with the leakage from the error terms of inner and outer query responses by adding noise flooding. Once again, we use LHL-based arguments to argue that the hints do not leak a significant amount of information, and add redundant randomness to ensure that the database satisfies the necessary min-entropy requirements. However, here the encoding of database elements using this randomness is done at two levels (the outer and inner level), so we use two random oracle hash functions $\mathsf{F}_{\mathsf{out}}$ and F_{in}. Essentially, our approach involves the server sampling random "dummy" databases $\{\mathbf{D}_k\}_{k \in [\omega_{\mathsf{out}} - 1]}$, and encrypting each outer database element $\mathbf{D}[i, j]$ by computing $\mathsf{F}_{\mathsf{out}}$ on all $\omega_{\mathsf{out}} - 1$ dummy elements in position (i, j). Each inner database is also encoded as $\left[\mathbf{H}^\top \mid \mathbf{B} \mid \overline{\mathbf{a}}\right]$, where $\mathbf{B}$ is sampled uniformly and $\overline{\mathbf{a}}$ is a row-wise encryption of $\mathbf{a}$ under $\mathsf{F}_{\mathsf{in}}(\mathbf{B})$.

The DoubleSPIR scheme. For simplicity, we consider database elements over $\mathbb{Z}_\rho$, each occupying an entry of $\mathsf{DB} \in \mathbb{Z}_\rho^{\sqrt{m} \times \sqrt{m}}$. The DoubleSPIR scheme is described as follows:

1. For $k \in [\omega_{\mathsf{out}} - 1]$, the server samples $\mathbf{D}_k \xleftarrow{\$} \mathbb{Z}_\rho^{\sqrt{m} \times \sqrt{m}}$, and sets $\mathbf{D}_{\omega_{\mathsf{out}}}$ to be the matrix such that
$$\mathbf{D}_{\omega_{\mathsf{out}}}[i,j] = \mathsf{DB}[i,j] + \mathsf{F}_{\mathsf{out}}(i, j, \mathbf{D}_1[i,j], \ldots, \mathbf{D}_{\omega_{\mathsf{out}}-1}[i,j]) \mod \rho.$$

2. For $k \in [\omega_{\mathsf{out}}]$, the server computes $\mathbf{H}_k = \mathbf{A} \cdot \mathbf{D}_k \in \mathbb{Z}_q^{n \times \sqrt{m}}$, with $\mathbf{A} \in \mathbb{Z}_q^{n \times \sqrt{m}}$ sampled (pseudo)randomly.

3. As public parameters, the server derives $\widehat{\mathbf{A}} \in \mathbb{Z}_q^{n \times \sqrt{m}}$ (pseudo)randomly and publishes
$$\left\{ \widehat{\mathbf{A}}, \mathbf{A}, \left\{ \widehat{\mathbf{A}} \cdot \overline{\mathbf{H}_k^\top} \right\}_{k \in \omega_{\mathsf{out}}} \right\},$$
where $\overline{\mathbf{H}^\top}$ denotes encoding each $\mathbb{Z}_q$ entry of $\mathbf{H}^\top$ as $\kappa = \log q / \log \rho$ blocks of size $\log \rho$, so that $\overline{\mathbf{H}^\top} \in \mathbb{Z}_\rho^{\sqrt{m} \times \kappa n}$.

   Note: in fact, the server need only publish the PRG seeds used to derive $\mathbf{A}$ and $\widehat{\mathbf{A}}$.

4. For each outer query, the server computes the answers $\mathbf{a}_1, \ldots, \mathbf{a}_{\omega_{\mathsf{out}}}$ as in SimplePIR while also adding flooding noise $\mathbf{e}_{\mathsf{flood}}$ from a discrete gaussian.

5. For each inner query, the server samples $\mathbf{B} \xleftarrow{\$} \mathbb{Z}_\rho^{\sqrt{m} \times \omega_{\mathsf{in}}}$. It additionally computes $\overline{\mathbf{a}'_k} \in \mathbb{Z}_\rho^{\sqrt{m} \times \kappa}$ for $k \in [\omega_{\mathsf{out}}]$, where $\mathbf{a}'_k[i] = \mathbf{a}_k[i] + \mathsf{F}_{\mathsf{in}}(i, \mathbf{B}[i, \cdot]) \mod q$ is an encryption of the k-th outer query answer.

   For for $k \in [\omega_{\mathsf{out}}]$, the server sets the k-th inner database as
$$\widehat{\mathbf{D}_k} = \left[\overline{\mathbf{H}_k^\top} \;\middle|\; \mathbf{B} \;\middle|\; \overline{\mathbf{a}'_k} \right] \in \mathbb{Z}_\rho^{\sqrt{m} \times (\kappa(n+1) + \omega_{\mathsf{in}})}.$$

   The server computes the inner response over each $\widehat{\mathbf{D}_k}$ as in SimplePIR, again adding noise flooding. Additionally, the server must send $\widehat{\mathbf{A}} \cdot \mathbf{B}$ and $\widehat{\mathbf{A}} \cdot \overline{\mathbf{a}'_k}$ for each $k \in \omega_{\mathsf{out}}$ to the client. Note that the *same* $\mathbf{B}$ is included in each of the ω_{out} inner databases $\widehat{\mathbf{D}_k}$ for ease of notation, but the portion of the response obtained by multiplying the inner query by $\mathbf{B}$ only needs to be computed and transmitted once.

We prove the database privacy of DoubleSPIR in the following two lemmas. In the result below, we consider as a "forbidden" input to F_{in} any $(j, \mathbf{B}[j, \cdot])$ for which the respective inner query was made for a different row j'. Intuitively, if no such forbidden query occurs, then the only leakage $\mathcal{A}$ sees comes from the *outer* hints and the *outer* query error terms, which simplifies the next step of our security proof.

Lemma 6.1. *Let ϕ be any database distribution. Let $\mathcal{A}$ be an adversary against the (ℓ, ϕ)-`sel-IND-DP` security game for the* DoubleSPIR *scheme making at most $q_{\mathsf{F}_{\mathsf{in}}}$ queries to F_{in} and whose queries are chosen independently of the database. For any $a \in (1, \infty)$, it holds that the probability that $\mathcal{A}$ queries F_{in} on a forbidden input is at most*
$$\left((\epsilon_1 \cdot q_{\mathsf{F}_{\mathsf{in}}} \cdot \ell + \epsilon_2) \cdot \exp\left(\frac{\ell \cdot a \cdot (\kappa(n+1)\omega_{\mathsf{out}} + \omega_{\mathsf{in}}) \cdot m\rho^2}{2\sigma^2} \right) \right)^{\frac{a-1}{a}},$$
where $\epsilon_1 = \left(\frac{\ell \omega_{\mathsf{in}}}{2} \sqrt{2^{n \log q - \sqrt{m} + \ell}} + 2^{-\omega_{\mathsf{in}} \log \rho} \right)$ and $\epsilon_2 = \sqrt{m} \cdot \ell^2 \cdot 2^{-\omega_{\mathsf{in}} \log \rho}$.

Proof. Let $\mathbf{G}^1$ be the game in which every inner error term $\mathbf{e}^\top \cdot \widehat{\mathbf{D}_k} + \mathbf{e}^\top_{\mathsf{flood}}$ is replaced by $\mathbf{e}^\top_{\mathsf{flood}}$, for every $k \in [\omega_{\mathsf{out}}]$ and for each of the ℓ queries. Let E_i denote the event that a forbidden query to F_{in} occurs in game $\mathbf{G}^i$.

Note that the trace of each game can be generated by an algorithm that receives as input the public parameters, all inner and outer database encodings, the adversary's random tape, and each of the ℓ inner query error terms $\left(\mathbf{e}^\top \cdot \widehat{\mathbf{D}_1} + \mathbf{e}^\top_{\mathsf{flood}_1}, \ldots, \mathbf{e}^\top \cdot \widehat{\mathbf{D}_{\omega_{\mathsf{out}}}} + \mathbf{e}^\top_{\mathsf{flood}_{\omega_{\mathsf{out}}}}\right)$ or $\left(\mathbf{e}^\top_{\mathsf{flood}_1}, \ldots, \mathbf{e}^\top_{\mathsf{flood}_{\omega_{\mathsf{out}}}}\right)$. By the same argument as before, we get that

$$\Pr[E_0] \leq \left(\Pr[E_1] \cdot \exp\left(\frac{\ell \cdot a \cdot (\kappa(n+1)\omega_{\mathsf{out}} + \omega_{\mathsf{in}}) \cdot m\rho^2}{2\sigma^2}\right)\right)^{\frac{a-1}{a}}.$$

Now, consider an algorithm $\mathcal{B}$ who receives $\widehat{\mathbf{A}}$, chooses indices $j_1, \ldots, j_\ell$, and gets $(\widehat{\mathbf{A}}, \widehat{\mathbf{A}} \cdot \mathbf{R}_1, \mathbf{R}_1[j_1, \cdot], \ldots, \widehat{\mathbf{A}} \cdot \mathbf{R}_\ell, \mathbf{R}_\ell[j_\ell, \cdot])$, where each $\mathbf{R}_t \xleftarrow{\$} \mathbb{Z}_\rho^{\sqrt{m} \times \omega_{\mathsf{in}}}$. $\mathcal{B}$'s goal is to guess a row $\mathbf{R}_t[j]$ with $j \neq j_t$.

Note that $\mathcal{B}$ can sample a database according to ϕ and generate the trace for game $\mathbf{G}^1$ by internally calling $\mathcal{A}$ as follows: $\mathcal{B}$ chooses its indices $j_1, \ldots, j_\ell$ to match $\mathcal{A}$'s indices for each inner query (note that these indices depend at most on $\mathbf{A}$ and $\widehat{\mathbf{A}}$). Thus, for each of the ℓ queries, $\mathcal{B}$ learns the corresponding rows of each $\mathbf{R}_t$. Indeed, this is sufficient for $\mathcal{B}$ to answer all of $\mathcal{A}$'s outer *and* inner queries as follows: to the t-th inner query for the k-th inner database $\mathbf{s}^\top \cdot \widehat{\mathbf{A}} + \mathbf{e}^\top + \mathbf{f}^\top$, $\mathcal{B}$ answers with

$$\mathbf{s}^\top \cdot \widehat{\mathbf{A}} \cdot \left[\overline{\mathbf{H}_k^\top} \;\middle|\; \mathbf{R}_t \;\middle|\; \mathbf{u}_t + \overline{\mathbf{a}_{t_k}}\right] + \mathbf{f}^\top \cdot \left[\overline{\mathbf{H}_k^\top} \;\middle|\; \mathbf{R}_t \;\middle|\; \mathbf{u}_t + \overline{\mathbf{a}_{t_k}}\right] + \mathbf{e}^\top_{\mathsf{flood}},$$

where $\mathbf{a}_{t_k}$ is the answer to the t-th outer query for the k-th outer database, and $\mathbf{u}_t$ is sampled uniformly by $\mathcal{B}$. Note here that the idea is to have $\mathcal{B}$ embed the matrices $\mathbf{R}_t$ into the inner database encodings of $\mathcal{A}$'s game.

Of course, $\mathcal{A}$ is also fed the necessary information to subtract the left hand side of the equations above, so they are able to retrieve the desired row of each inner matrix. When $\mathcal{A}$ queries F_{in} on $(j_t, \mathbf{R}_t[j_t, \cdot])$, $\mathcal{B}$ responds to the query with $\mathbf{u}_t[j_t]$. For other queries, $\mathcal{B}$ responds with fresh random values.

Notice that, as long as all $\mathbf{R}_1[j, \cdot], \ldots, \mathbf{R}_\ell[j, \cdot]$ are different for each fixed $j \in [\sqrt{m}]$ (which holds with probability at least $1 - \sqrt{m} \cdot \binom{\ell}{2} \cdot 2^{-\omega_{\mathsf{in}} \log \rho}$, ensuring consistency of the values $\mathcal{B}$ implicitly defines as random oracle outputs), then $\mathcal{B}$ is effectively simulating game $\mathbf{G}^1$, provided that $\mathcal{A}$ hasn't queried F_{in} on a forbidden input. If such a forbidden query occurs, then $\mathcal{B}$ picks $t \xleftarrow{\$} [\ell]$, and picks one of $\mathcal{A}$'s $q_{\mathsf{F}_{\mathsf{in}}}$ random oracle query inputs $(j, \mathbf{q})$ at random. $\mathcal{B}$ then guesses that $\mathbf{R}_t[j] = \mathbf{q}$. In this case, $\mathcal{B}$ wins with probability at least $1/(\ell \cdot q_{\mathsf{F}_{\mathsf{in}}})$.

By our generalized LHL, we know that

$$\mathsf{SD}((\widehat{\mathbf{A}}, \widehat{\mathbf{A}} \cdot \mathbf{R}_1, \mathbf{R}_1[j_1, \cdot], \ldots, \widehat{\mathbf{A}} \cdot \mathbf{R}_\ell, \mathbf{R}_\ell[j_\ell, \cdot]), (\widehat{\mathbf{A}}, \mathbf{U}_1, \mathbf{R}_1[j_1, \cdot], \ldots, \mathbf{U}_\ell, \mathbf{R}_\ell[j_\ell, \cdot])) \leq \frac{\ell\omega_{\mathsf{in}}}{2}\sqrt{2^{n \log q - \sqrt{m} + \ell}},$$

so $\mathcal{B}$'s advantage can be at most $\frac{\ell\omega_{\mathsf{in}}}{2}\sqrt{2^{n \log q - \sqrt{m} + \ell}} + 2^{-\omega_{\mathsf{in}} \log \rho}$. Thus, we conclude that

$$\Pr[E_1] \leq \left(\frac{\ell\omega_{\mathsf{in}}}{2}\sqrt{2^{n \log q - \sqrt{m} + \ell}} + 2^{-\omega_{\mathsf{in}} \log \rho}\right) \cdot q_{\mathsf{F}_{\mathsf{in}}} \cdot \ell + \sqrt{m} \cdot \ell^2 \cdot 2^{-\omega_{\mathsf{in}} \log \rho}.$$

□

Below, we consider "forbidden" queries to $\mathsf{F}_{\mathsf{out}}$ as being any query on an input $(i, j, \mathbf{D}_1[i, j], \ldots, \mathbf{D}_{\omega_{\mathsf{out}}-1}[i, j])$, where (i, j) is a database position which is not queried by $\mathcal{A}$. Note that, if $\mathcal{A}$ makes no forbidden outer queries, then every unqueried element is masked by a uniform value. Thus, database privacy follows immediately by upper bounding the probability that forbidden outer queries occur.

Lemma 6.2. *Let ϕ be any database distribution. Let $\mathcal{A}$ be an adversary against the (ℓ, ϕ)-`sel-IND-DP` security game for the* DoubleSPIR *scheme making at most $q_{\mathsf{F}_{\mathsf{out}}}$ queries to $\mathsf{F}_{\mathsf{out}}$ and whose queries are chosen independently of the database. For any $a \in (1, \infty)$, it holds that the probability that $\mathcal{A}$ queries $\mathsf{F}_{\mathsf{out}}$ on a forbidden input, conditioned on the fact that $\mathcal{A}$ does not query F_{in} on a forbidden input, is at most*

$$\left(q_{\mathsf{F}_{\mathsf{out}}} \cdot \left(\epsilon + 2^{-(\omega_{\mathsf{out}}-1)\log\rho}\right) \cdot \exp\left(\frac{\ell \cdot \omega_{\mathsf{out}} \cdot a \cdot \sqrt{m} \cdot m\rho^2}{2\sigma^2}\right)\right)^{\frac{a-1}{a}},$$

where $\epsilon = \frac{\omega_{\mathsf{out}}\sqrt{m}}{2}\sqrt{2^{n\log q - \sqrt{m}+\ell}}$.

Proof. By assumption, all positions in the outer query answers are completely masked from $\mathcal{A}$ (except for the desired one), since no forbidden inner queries are made. This means that the only leakage $\mathcal{A}$ sees is due to the outer hints $\mathbf{H}_1, \ldots, \mathbf{H}_{\omega_{\mathsf{out}}}$ and due to the error term from the outer queries.

We can use the same Rényi noise flooding argument as before to prove that

$$\Pr[E_0] \leq \left(\Pr[E_1] \cdot \exp\left(\frac{\ell \cdot \omega_{\mathsf{out}} \cdot a \cdot \sqrt{m} \cdot m\rho^2}{2\sigma^2}\right)\right)^{\frac{a-1}{a}},$$

where $\mathbf{G}^1$ is defined as the game where the leakage error term is removed and only the flooding noise is included in *outer* query responses.

All that is left is to deal with the leakage from the hints $\mathbf{H}_1, \ldots, \mathbf{H}_{\omega_{\mathsf{out}}}$. To this end, we construct $\mathcal{B}$ who receives $\mathbf{A}$, chooses ℓ positions $(i_1, j_1), \ldots, (i_\ell, j_\ell)$, gets $\{(\mathbf{A}\cdot\mathbf{R}_k, \mathbf{R}_k[i_1, j_1], \ldots, \mathbf{R}_k[i_\ell, j_\ell])\}_{k\in[\omega_{\mathsf{out}}]}$ (with each $\mathbf{R}_k \xleftarrow{\$} \mathbb{Z}_\rho^{\sqrt{m}\times\sqrt{m}}$), and attempts to guess $\mathbf{R}_1[i, j], \ldots, \mathbf{R}_{\omega_{\mathsf{out}}-1}[i, j]$, for some $(i, j) \neq (i_k, j_k)$ for every $k \in [\omega_{\mathsf{out}} - 1]$.

$\mathcal{B}$ chooses the same query ℓ indices as $\mathcal{A}$. $\mathcal{B}$ can then simulate $\mathcal{A}$'s game as follows: $\mathcal{B}$ samples the database DB from the appropriate distribution, then sets the outer matrices $\mathbf{D}_1, \ldots, \mathbf{D}_{\omega_{\mathsf{out}}}$ as $\mathbf{R}_1, \ldots, \mathbf{R}_{\omega_{\mathsf{out}}}$ (note that $\mathbf{R}_1, \ldots, \mathbf{R}_{\omega_{\mathsf{out}}}$ are unknown to $\mathcal{B}$, so $\mathcal{B}$ must perform the simulation using only partial knowledge about these matrices); when $\mathcal{A}$ queries position (i_t, j_t) for some $t \in [\ell]$, $\mathcal{B}$ computes the j_t-th entry of each of the outer query answers $\mathbf{a}_1, \ldots, \mathbf{a}_{\omega_{\mathsf{out}}}$ (and subsequently of $\mathbf{a}'_1, \ldots, \mathbf{a}'_{\omega_{\mathsf{out}}}$) as specified by the protocol, and the remaining entries of each $\mathbf{a}'_k$ are sampled uniformly at random (since $\mathcal{B}$ does not necessarily know $\mathbf{R}_k[i_t, j']$ for $j' \neq j_t$). Moreover, when $\mathcal{A}$ queries $\mathsf{F}_{\mathsf{out}}$ on $(i_t, j_t, \mathbf{R}_1[i_t, j_t], \ldots, \mathbf{R}_{\omega_{\mathsf{out}}-1}[i_t, j_t])$, $\mathcal{B}$ responds to the query with the value $\mathbf{R}_{\omega_{\mathsf{out}}}[i_t, j_t] + \mathsf{DB}[i_t, j_t] \bmod \rho$ (so that the value retrieved by $\mathcal{A}$ is consistent with the database distribution ϕ).

Note that $\mathcal{A}$'s view inside $\mathcal{B}$'s simulation is identical to the view in $\mathbf{G}^1$ as long as $\mathcal{A}$ doesn't query $\mathsf{F}_{\mathsf{out}}$ on forbidden inputs. This is because $\mathcal{A}$ does not query F_{in} on forbidden inner inputs, meaning that unqueried entries for the outer answers are indeed uniformly masked. Thus, if a forbidden query by $\mathcal{A}$ occurs then $\mathcal{B}$ wins with probability $1/q_{\mathsf{F}_{\mathsf{out}}}$ by simply selecting one of $\mathcal{A}$'s queries to $\mathsf{F}_{\mathsf{out}}$ at random. However, by the generalized LHL, $\mathcal{B}$'s input has statistical distance at most

$$\epsilon = \frac{\omega_{\mathsf{out}}\sqrt{m}}{2}\sqrt{2^{n\log q - \sqrt{m}+\ell}}$$

from $\{(\mathbf{A}, \mathbf{U}_k, \mathbf{R}_k[i_1, j_1], \ldots, \mathbf{R}_k[i_\ell, j_\ell])\}_{k \in \omega_{\mathsf{out}}}$, where each $\mathbf{U}_k \xleftarrow{\$} \mathbb{Z}_q^{n \times \sqrt{m}}$. This means that $\mathcal{B}$'s advantage is at most $\epsilon + 2^{-(\omega_{\mathsf{out}}-1)\log\rho}$, so $\mathcal{A}$'s advantage in $\mathbf{G}^1$ is upper bounded by

$$q_{\mathsf{F}_{\mathsf{out}}} \cdot (\epsilon + 2^{-(\omega_{\mathsf{out}}-1)\log\rho}).$$

□

Using the two lemmas above, we are now ready to prove the selective flavor of indistinguishability-based database privacy of DoubleSPIR, with adaptive privacy again following from Theorem 3.6.

Theorem 6.3 ((ℓ, Φ)-sel-IND-DP security)**.** *Let ϕ be any database distribution. Let $\mathcal{A}$ be an adversary against the (ℓ, ϕ)-sel-IND-DP security game for the* DoubleSPIR *scheme and whose queries are chosen independently of the database. Then*

$$\mathsf{Adv}^{(\ell,\phi)\text{-sel-IND-DP}}(\mathcal{A}) \leq p_{\mathsf{F}_{\mathsf{in}}} + p_{\mathsf{F}_{\mathsf{out}}},$$

where $p_{\mathsf{F}_{\mathsf{in}}}, p_{\mathsf{F}_{\mathsf{out}}}$ are the bounds from Lemma 6.1 and Lemma 6.2, respectively.

Proof. The probability that $\mathcal{A}$ makes a forbidden outer query is at most the probability that $\mathcal{A}$ makes a forbidden inner query plus the probability that $\mathcal{A}$ makes a forbidden outer query conditioned on the fact that $\mathcal{A}$ does *not* make a forbidden inner query. As long as $\mathcal{A}$ makes no forbidden outer queries, their view in the security game is independent of unqueried database elements, and the result follows. □

Corollary 6.4 ((ℓ, Φ)-IND-DP security)**.** *Let ϕ be any database distribution. Let $\mathcal{A}$ be an adversary against the (ℓ, ϕ)-IND-DP security game for the* DoubleSPIR *scheme. Then*

$$\mathsf{Adv}^{(\ell,\phi)\text{-IND-DP}}(\mathcal{A}) \leq \binom{m}{\ell} \cdot (p_{\mathsf{F}_{\mathsf{in}}} + p_{\mathsf{F}_{\mathsf{out}}}),$$

where $p_{\mathsf{F}_{\mathsf{in}}}, p_{\mathsf{F}_{\mathsf{out}}}$ are the bounds from Lemma 6.1 and Lemma 6.2, respectively.

Proof. Immediate from Theorem 3.6 and Theorem 6.3. □

7 Parameters and Performance Evaluation

We focus on our $\mathsf{SPIR}^{\mathsf{ro}}$ scheme based on FrodoPIR. The smallest power-of-two modulus we are able to achieve while maintaining correctness and security is $q = 2^{64}$, which still supports native hardware arithmetic. The LWE dimension is estimated using the estimator of [2]. For the LWE secret and error distributions considered, it is estimated that choosing the LWE dimension $n = 2520$ is sufficient to provide 128 bit security for plain decisional LWE for all attacks covered by the online estimator.

Comparison to statistical distance analysis. We remark that replacing the Rényi divergence by the statistical distance in our security proofs yields an *additive* term significantly larger than $1/\sigma$ in the loss, which implies that σ must be of magnitude 2^λ to ensure λ bits of security. For correctness to hold with probability at least $1 - 2^{-32}$, recall that the equation $q > 16\rho\sigma$ must hold. This imposes that a modulus $q \gg 2^{128}$ would be needed, leading to a significant worsening of scheme parameters.

Moreover, we recall that our reduction from adaptive to selective security involves a *multiplicative* loss term of $\binom{m}{\ell}$. Thus, for adaptive security it would be required that $\sigma \gg \binom{m}{\ell}$ when using

the statistical distance analysis. For the practical values of m and ℓ we consider, the resulting modulus size is in the kilobyte range, and thus unacceptable. This highlights the usefulness of the transformations described in Section 5, as security improves exponentially in the number of extra bits added to each database element and it is not necessary for the modulus to tolerate very large noise.

Adaptive queries. Below we suggest concrete parameters to be used in our $\mathsf{SPIR}^{\mathsf{ro}}$ scheme in a way that guarantees correctness with probability at least $1 - 2^{-32}$ and that the advantage of any adversary against our adaptive security game is at most $2^{-\lambda}$. For the sake of simplicity, we ignore the loss that derived from using a pseudorandom $\mathbf{A}$ instead of truly random - this can be compensated for by using a PRG seed as large as necessary. Thus, by Theorem 5.1, and taking into account the additional $\log \binom{m}{\ell} \leq \ell \cdot \log(\frac{em}{\ell})$ adaptive loss due to Theorem 3.6, it is *approximately* sufficient to ensure that

$$\lambda \lesssim \frac{a-1}{a}\left(s - \log_2(e)\frac{\ell\omega' a(m\rho)^2}{2\sigma^2}\right) - \ell \cdot \log\left(\frac{em}{\ell}\right) - \log(q_{\mathsf{F}}),$$

if m is large enough to ensure $\sqrt{2^{n\log q - m + \ell + 1}}$ is insignificant (e.g., $m \geq 2^{18}$). Moreover, choosing σ large enough so that $\ell\omega' a(m\rho)^2 = c\sigma^2$ for some sufficiently small constant c, bounding the number of database queries at $\ell := \sqrt{m}$ and setting the order of the Rényi divergence a to be large enough, we see that it is sufficient to set s slightly larger than $\lambda + \sqrt{m}\log(e\sqrt{m})$.

In Table 1 we omit the parameter q_{F} as for all plausible values, $\log(q_{\mathsf{F}}) \ll \ell \log\left(\frac{em}{\ell}\right)$. We have $\eta := (m\log(\rho)\cdot\omega)/(n\log(q)\cdot\omega')$ denoting the improvement factor for offline client download relative to the original database size, and $\tau = (w+s)/w$ denoting the blow-up in the size of the database, after enlarging each entry with s-bit keys. While it is important to reduce offline client download, we note that having client download close to, or even surpassing, the original database size does not render an SPIR protocol redundant, in contrast to the PIR setting. All values provide at least 128 bit security against adaptive adversaries. However, for simplicity we conservatively overestimate the value of s by using the nearest sufficiently large power of 2, which typically results in much higher security.

m	ℓ	n	ρ	σ	w	a	s	η	τ
2^{18}	2^{9}	2520	2^{16}	2^{43}	2^{13}	2^{10}	2^{13}	9.49	2
2^{20}	2^{10}	2520	2^{14}	2^{45}	2^{13}	2^{14}	2^{14}	22.16	3
2^{24}	2^{12}	2520	2^{12}	2^{47}	2^{13}	2^{10}	2^{15}	182.36	5

Table 1: Examples of concrete parameters for $\mathsf{SPIR}^{\mathsf{ro}}$.

The hint download relative to database size decreases as m grows, since the hint in FrodoPIR is independent of m. On the other hand, due to the significant loss in the reduction from adaptive to selective security, the server must add redundancy to the database relative to its original size for larger values of m.

7.1 Performance Evaluation

We provide an experimental analysis of the incurred computational runtimes and bandwidth usage when running SPIR and $\mathsf{SPIR}^{\mathsf{ro}}$, as compared with the original FrodoPIR scheme used as the basis for these schemes.

Experimental Setup.

We run all experiments as single-threaded processes on cluster nodes running Debian 12, equipped with dual 16-core AMD EPYC 7343 processors and 128 GB RAM. All computational costs correspond to CPU processing time. Our open-source implementation modifies the FrodoPIR implementation directly.[6]

Parameter choices. We provide non-amortized communication and computation benchmarks for a single server database using respective parameter settings dependent on database size (m) and scheme for both $m = 2^{18}$ and $m = 2^{20}$. The parameters used for SPIR/SPIR$^{\mathsf{ro}}$ can be found in Table 1, and provide 128-bit security for l client queries.

We use $w = 2^{13}$ (i.e., 1KB database elements) and $n = 2520$. Moreover, the random oracle F used in SPIR$^{\mathsf{ro}}$ is instantiated as the SHAKE256 [55] extendable output function.

For FrodoPIR, we use parameters from the original paper [20]: $n = 1774$, $\rho = 10$ for $m = 2^{18}$, and $\rho = 9$ for $m = 2^{20}$. Additionally the modulus q in FrodoPIR is 2^{32} while our schemes use $q = 2^{64}$.

Results.

Table 2 displays the non-amortized performance comparison between the different schemes. This involves calculating running times and bandwidth usage for running a single-threaded server instance in interaction with a single client. While these evaluations focus on the (non-amortized) performance of the schemes relative to each other, the computational and communication costs can be similarly amortized as demonstrated in the original FrodoPIR work, which would yield comparable improvements for SPIR and SPIR$^{\mathsf{ro}}$.

Offline phase. Across all schemes, the server generates their database matrix DB and public parameters. This server-side operation scales linearly with m for both FrodoPIR and SPIR, and additionally with s for the SPIR$^{\mathsf{ro}}$ scheme, which explains the higher runtimes regarding database processing. This process includes the pseudorandom derivation of the LWE matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, measured separately to assess its runtime for a given client. The cost of the derivation is predictably higher in SPIR/SPIR$^{\mathsf{ro}}$ which use an LWE dimension that is roughly twice that of FrodoPIR. The increase in LWE dimension is also the main cause for the longer query-independent pre-processing times on the client-side for these schemes. Ultimately, these costs still grow roughly linearly in m across all schemes.

Client download increases in SPIR/SPIR$^{\mathsf{ro}}$ relative to FrodoPIR due to larger parameters being used in general (n, ω and q). This is especially noticeable in SPIR$^{\mathsf{ro}}$ due to s being factored in the calculation of ω.

Online phase. The client computation consists of performing a single addition operation to modify a single portion of preprocessed data. These times only differ noticeably between the SPIR/SPIR$^{\mathsf{ro}}$ schemes and FrodoPIR due to an implementation detail in which the method for generating the query clones the b array. Since the b array consists of 64-bit elements for our schemes, cloning it takes approximately twice as long. Server responses require post-processing by the client. These times scale linearly with ω for FrodoPIR and SPIR. SPIR$^{\mathsf{ro}}$ requires additional XOR and hashing operations which additionally depend on s, and lead to increased computation times. Server-side

[6] SPIR: https://anonymous.4open.science/r/spir-1C75
SPIR$^{\mathsf{ro}}$: https://anonymous.4open.science/r/spir_ro-6BFB

Scheme	FrodoPIR		SPIR		$\mathsf{SPIR}^{\mathsf{ro}}$	
DB Items (log(m))	18	20	18	20	18	20
Hint processing (s)	139	569	206	881	402	2665
Hint download (KB)	5682	6313	10080	11537	20160	34571
Query computation (s)	0.17943	0.68389	0.40633	1.5688	0.39340	1.5930
Resp. computation (ms)	57.843	232.69	67.277	302.60	145.96	925.39
Client decode (ms)	0.16911	0.18643	0.15572	0.17841	0.48290	0.81291
Query size (KB)	1024	4096	2048	8192	2048	8192
Resp. size (KB)	3.203	3.559	4	4.578	8	13.719

Table 2: Non-amortized computation and bandwidth costs for database sizes of 2^{18} and 2^{20} comparing standard FrodoPIR, with the SPIR variant.

query processing runtime differences largely hinge on the size of w, which is highest in $\mathsf{SPIR}^{\mathsf{ro}}$. Although SPIR runs use values of ω lower than those in FrodoPIR (e.g., 512 vs. 820), the additional computational costs can be attributed to the noise flooding introduced in the computation.

Client query communication costs double from FrodoPIR to our schemes due to the modulus increase. The server response is also larger for both our schemes due to increases in the modulus (and ω for $\mathsf{SPIR}^{\mathsf{ro}}$).

Comparison with OPRF-based PIR scheme. In Section 8, we provide a performance (and security) comparison of our base SPIR scheme (SPIR) with an SPIR scheme instantiated using standard FrodoPIR and a state-of-the-art post-quantum (and semi-honest) OPRF protocol (Pool [19]). We find that high interactive offline costs for the OPRF scheme ensure that our approach provides a simplified and streamlined interaction for achieving database privacy.

8 Comparison with OPRF-based SPIR

As noted in the introduction, in OPRF-based SPIR schemes, the server encrypts the i^{th} database row under a secret key derived as $\mathsf{k}_i \leftarrow \mathsf{F}(\mathsf{sk}, i)$ (where F is a pseudorandom function, and sk is a server-held key). Then, in order to query the i^{th} row, the client sends a PIR query for the i^{th} index, along with a parallel OPRF query to learn k_i. On receiving both responses, the client decrypts the encrypted entry it receives, and outputs the contents it receives. Intuitively, database privacy follows from the fact that the client must send OPRF queries for each row it wants to learn, and it cannot learn any rows for which it does not send an OPRF query, by the IND-CPA security (or otherwise) of the encryption system that is used.

Security. Note that, conceptually, there are still some gaps in how to define the security guarantee provided. In particular, a malicious client could query for different indices in the PIR and OPRF protocols which may allow it to recover certain unintended database entries, even when utilising a malicious OPRF. In particular, without noise flooding in LWE-based PIR, it is theoretically possible that the noise in query responses may allow clients to leak encrypted entries from the database, without ever querying them directly using the PIR protocol. They could then run OPRF queries for said rows to decrypt them, which would appear to invalidate the database privacy property somewhat. While this would appear to be a pathological case, it illustrates that running parallel functionalities can lead to uncaptured edge cases in security notions.

Performance comparison. In Section 7, we see that our transformation leads to 2× and 4× performance overheads with respect to communication and computation, respectively, as compared with the original LWE-based PIR scheme (FrodoPIR). Note that FrodoPIR overloads the client query (resulting in a size of 8 MB in SPIR for a database of 2^{20} elements), while keeping server responses very small (~ 4.5 kB in SPIR). As mentioned in Section 1, this reduces financial costs in commercial deployments of the PIR scheme, where uploads are free and downloads are expensive. In the offline phase, the client hint download corresponds to around 11 MB for the same database size. Server response runtimes are in the orders of milliseconds.

To compare briefly with the costs of running standard FrodoPIR with an OPRF, we first note that to maintain post-quantum security (which we achieve in our schemes) we should utilise post-quantum OPRFs. In particular, the OPRF with state-of-the-art performance that is based on a well-known post-quantum hardness assumption would appear to be Pool [19]. While Pool only guarantees semi-honest security, this would appear to be sufficient in the PIR case (which is semi-honest by default). In Pool, each client and server must run an interactive offline phase, before subsequently running a cheaper online phase. The protocol is based on the usage oblivious transfer (OT), when instantiated with the LWE-based OT scheme known as KyberOT [43].

To summarise performance, we first rerun the benchmarks provided in [19] (using $\lambda = 128$, where $n = 482$, $p = 2^8$, $q = 2^{12}$ from the Pool parameter set), and permitting 2^{13} evaluations. Our measurements show single-threaded online runtimes of 528 µs (client-side) and 368 µs (server-side). Offline computation via IKNP OT extension takes 370 ms (client) and 167 ms (server). In the case of Silent OT, these values are approximately 2 s in both cases.[7] Secondly, looking at bandwidth, the Pool offline phase (when using IKNP OT extension) requires 100 MB of interactive communication between the client and server, before consuming around 12 kB of online communication for online queries. This offline communication can be reduced to around 1 MB, but at the cost of the interaction taking much longer (over 2 seconds).

Overall, the offline costs of running the Pool OPRF alongside the standard instantiation of FrodoPIR would appear to singificantly outweigh the costs required by SPIR. Furthermore, since the offline phase is interactive and per-client, the server computation does not amortise over all clients globally. In SPIR, server offline computation is restricted to a single matrix multiplication, that clients can download non-interactively, which provides much less of a burden on the server (and on the establishment of network connections, etc.). Overall, we believe SPIR provides a much cleaner interface for establishing database privacy in PIR schemes, both from a security and a performance perspective.

References

[1] Agrawal, S., Stehlé, D., Yadav, A.: Round-optimal lattice-based threshold signatures, revisited. In: Bojańczyk, M., Merelli, E., Woodruff, D.P. (eds.) 49th International Colloquium on Automata, Languages, and Programming (ICALP 2022). Leibniz International Proceedings in Informatics (LIPIcs), vol. 229, pp. 8:1–8:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany (2022). https://doi.org/10.4230/LIPIcs.ICALP.2022.8

[2] Albrecht, M., Player, R., Scott, S.: On the concrete hardness of learning with errors. Journal of Mathematical Cryptology **9** (10 2015). https://doi.org/10.1515/jmc-2015-0016

[3] Albrecht, M.R., Davidson, A., Deo, A., Gardham, D.: Crypto dark matter on the torus - oblivious PRFs from shallow PRFs and TFHE. In: Joye, M., Leander, G. (eds.) EU-

[7] This is around 3-4x quicker than the originally reported numbers in [19].

ROCRYPT 2024, Part VI. LNCS, vol. 14656, pp. 447–476. Springer, Cham (May 2024). https://doi.org/10.1007/978-3-031-58751-1$_1$6

[4] Albrecht, M.R., Gür, K.D.: Verifiable oblivious pseudorandom functions from lattices: Practical-ish and thresholdisable. In: Chung, K.M., Sasaki, Y. (eds.) ASIACRYPT 2024, Part IV. LNCS, vol. 15487, pp. 205–237. Springer, Singapore (Dec 2024). https://doi.org/10.1007/978-981-96-0894-2$_7$

[5] Ali, A., Lepoint, T., Patel, S., Raykova, M., Schoppmann, P., Seth, K., Yeo, K.: Communication–Computation trade-offs in PIR. In: 30th USENIX Security Symposium (USENIX Security 21). pp. 1811–1828. USENIX Association (Aug 2021), https://www.usenix.org/conference/usenixsecurity21/presentation/ali

[6] Apple Inc.: Combining Machine Learning and Homomorphic Encryption in the Apple Ecosystem. Web page (2024), https://machinelearning.apple.com/research/homomorphic-encryption, accessed: November 2025

[7] Bai, S., Lepoint, T., Roux-Langlois, A., Sakzad, A., Stehlé, D., Steinfeld, R.: Improved security proofs in lattice-based cryptography: Using the Rényi divergence rather than the statistical distance. Journal of Cryptology **31**(2), 610–640 (2018). https://doi.org/10.1007/s00145-017-9265-9

[8] Beimel, A., Ishai, Y., Malkin, T.: Reducing the servers computation in private information retrieval: PIR with preprocessing. In: Advances in Cryptology - CRYPTO 2000, 20th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2000, Proceedings. Lecture Notes in Computer Science, vol. 1880, pp. 55–73. Springer (2000). https://doi.org/10.1007/3-540-44598-6$_4$

[9] Boudgoust, K., Costache, A.: Improved Rényi arguments for lattice-based threshold encryption. Cryptology ePrint Archive, Paper 2025/735 (2025), https://eprint.iacr.org/2025/735

[10] Boudgoust, K., Lapiha, O.: Leftover hash lemma(s) over cyclotomic rings. Cryptology ePrint Archive, Paper 2025/1080 (2025), https://eprint.iacr.org/2025/1080

[11] Boudgoust, K., Scholl, P.: Simple threshold (fully homomorphic) encryption from LWE with polynomial modulus. In: Guo, J., Steinfeld, R. (eds.) ASIACRYPT 2023, Part I. LNCS, vol. 14438, pp. 371–404. Springer, Singapore (Dec 2023). https://doi.org/10.1007/978-981-99-8721-4$_1$2

[12] Canonne, C.L., Kamath, G., Steinke, T.: The discrete gaussian for differential privacy. In: Proceedings of the 34th International Conference on Neural Information Processing Systems. NIPS '20, Curran Associates Inc., Red Hook, NY, USA (2020)

[13] Celi, S., Davidson, A.: Call me by my name: Simple, practical private information retrieval for keyword queries. In: Luo, B., Liao, X., Xu, J., Kirda, E., Lie, D. (eds.) ACM CCS 2024. pp. 4107–4121. ACM Press (Oct 2024). https://doi.org/10.1145/3658644.3670271

[14] Chor, B., Goldreich, O., Kushilevitz, E., Sudan, M.: Private information retrieval. In: Proceedings of IEEE 36th Annual Foundations of Computer Science. pp. 41–50 (1995). https://doi.org/10.1109/SFCS.1995.492461

[15] Chor, B., Gilboa, N., Naor, M.: Private information retrieval by keywords. Cryptology ePrint Archive, Paper 1998/003 (1998), https://eprint.iacr.org/1998/003

[16] Chowdhury, S., Sinha, S., Singh, A., Mishra, S., Chaudhary, C., Patranabis, S., Mukherjee, P., Chatterjee, A., Mukhopadhyay, D.: Efficient threshold FHE for privacy-preserving applications. Cryptology ePrint Archive, Paper 2022/1625 (2022), https://eprint.iacr.org/2022/1625

[17] Corrigan-Gibbs, H., Henzinger, A., Kogan, D.: Single-server private information retrieval with sublinear amortized time. In: Dunkelman, O., Dziembowski, S. (eds.) EUROCRYPT 2022, Part II. LNCS, vol. 13276, pp. 3–33. Springer, Cham (May / Jun 2022). https://doi.org/10.1007/978-3-031-07085-3$_1$

[18] Corrigan-Gibbs, H., Kogan, D.: Private information retrieval with sublinear online time. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Part I. LNCS, vol. 12105, pp. 44–75. Springer, Cham (May 2020). https://doi.org/10.1007/978-3-030-45721-1$_3$

[19] Davidson, A., Deo, A., Thibault, L.T.: Pool: A practical OT-based OPRF from learning with rounding. In: Huang, C., Chen, J., Shieh, S., Lie, D., Cortier, V. (eds.) Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security, CCS 2025, Taipei, Taiwan, October 13-17, 2025. pp. 1038–1052. ACM (2025). https://doi.org/10.1145/3719027.3765054

[20] Davidson, A., Pestana, G., Celi, S.: FrodoPIR: Simple, scalable, single-server private information retrieval. In: Proceedings on Privacy Enhancing Technologies. pp. 365–383. Privacy Enhancing Technologies Symposium Advisory Board (2023)

[21] Di Crescenzo, G., Malkin, T., Ostrovsky, R.: Single database private information retrieval implies oblivious transfer. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 122–138. Springer, Berlin, Heidelberg (May 2000). https://doi.org/10.1007/3-540-45539-6$_1$0

[22] van Erven, T., Harremoës, P.: Rényi divergence and Kullback-Leibler divergence. IEEE Transactions on Information Theory **60**, 3797–3820 (2012), https://api.semanticscholar.org/CorpusID:17522805

[23] Freedman, M.J., Ishai, Y., Pinkas, B., Reingold, O.: Keyword search and oblivious pseudorandom functions. In: Proceedings of the Second International Conference on Theory of Cryptography. p. 303–324. TCC'05, Springer-Verlag, Berlin, Heidelberg (2005). https://doi.org/10.1007/978-3-540-30576-7$_1$7

[24] Freedman, M.J., Nissim, K., Pinkas, B.: Efficient private matching and set intersection. In: Cachin, C., Camenisch, J. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 1–19. Springer, Berlin, Heidelberg (May 2004). https://doi.org/10.1007/978-3-540-24676-3$_1$

[25] Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. In: Wiener, M.J. (ed.) CRYPTO'99. LNCS, vol. 1666, pp. 537–554. Springer, Berlin, Heidelberg (Aug 1999). https://doi.org/10.1007/3-540-48405-1$_3$4

[26] Gertner, Y., Ishai, Y., Kushilevitz, E., Malkin, T.: Protecting data privacy in private information retrieval schemes. In: Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing. p. 151–160. STOC '98, Association for Computing Machinery, New York, NY, USA (1998). https://doi.org/10.1145/276698.276723

[27] Gil, M., Alajaji, F., Linder, T.: Rényi divergence measures for commonly used univariate continuous distributions. Information Sciences **249**, 124–131 (11 2013). https://doi.org/10.1016/j.ins.2013.06.018

[28] Green, M., Ladd, W., Miers, I.: A protocol for privately reporting ad impressions at scale. In: Weippl, E.R., Katzenbeisser, S., Kruegel, C., Myers, A.C., Halevi, S. (eds.) ACM CCS 2016. pp. 1591–1601. ACM Press (Oct 2016). https://doi.org/10.1145/2976749.2978407

[29] Hao, M., Liu, W., Peng, L., Zhang, C., Wu, P., Zhang, L., Li, H., Deng, R.H.: Practical keyword private information retrieval from key-to-index mappings. In: Proceedings of the 34th USENIX Conference on Security Symposium. SEC '25, USENIX Association, USA (2025)

[30] Heimberger, L., Kales, D., Lolato, R., Mir, O., Ramacher, S., Rechberger, C.: Leap: A fast, lattice-based OPRF with application to private set intersection. In: Fehr, S., Fouque, P.A. (eds.) EUROCRYPT 2025, Part VII. LNCS, vol. 15607, pp. 254–283. Springer, Cham (May 2025). https://doi.org/10.1007/978-3-031-91098-$2_1$0

[31] Henzinger, A., Dauterman, E., Corrigan-Gibbs, H., Zeldovich, N.: Private web search with tiptoe. In: Proceedings of the 29th Symposium on Operating Systems Principles. p. 396–416. SOSP '23, Association for Computing Machinery, New York, NY, USA (2023). https://doi.org/10.1145/3600006.3613134

[32] Henzinger, A., Hong, M.M., Corrigan-Gibbs, H., Meiklejohn, S., Vaikuntanathan, V.: One server for the price of two: Simple and fast Single-Server private information retrieval. In: 32nd USENIX Security Symposium (USENIX Security 23). pp. 3889–3905. USENIX Association, Anaheim, CA (Aug 2023), https://www.usenix.org/conference/usenixsecurity23/presentation/henzinger

[33] Jarecki, S., Jutla, C.S., Krawczyk, H., Rosu, M.C., Steiner, M.: Outsourced symmetric private information retrieval. In: Sadeghi, A.R., Gligor, V.D., Yung, M. (eds.) ACM CCS 2013. pp. 875–888. ACM Press (Nov 2013). https://doi.org/10.1145/2508859.2516730

[34] Kogan, D., Corrigan-Gibbs, H.: Private blocklist lookups with checklist. In: 30th USENIX Security Symposium (USENIX Security 21). pp. 875–892. USENIX Association (Aug 2021), https://www.usenix.org/conference/usenixsecurity21/presentation/kogan

[35] Kushilevitz, E., Ostrovsky, R.: Replication is not needed: single database, computationally-private information retrieval. In: Proceedings of the 38th Annual Symposium on Foundations of Computer Science. p. 364. FOCS '97, IEEE Computer Society, USA (1997)

[36] Langlois, A., Stehlé, D., Steinfeld, R.: GGHLite: More efficient multilinear maps from ideal lattices. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 239–256. Springer, Berlin, Heidelberg (May 2014). https://doi.org/10.1007/978-3-642-55220-$5_1$4

[37] Layouni, M.: Accredited symmetrically private information retrieval. In: Proceedings of the Security 2nd International Conference on Advances in Information and Computer Security. p. 262–277. IWSEC'07, Springer-Verlag, Berlin, Heidelberg (2007)

[38] Li, B., Micciancio, D., Raykova, M., Schultz, M.: Hintless single-server private information retrieval. In: Reyzin, L., Stebila, D. (eds.) CRYPTO 2024, Part IX. LNCS, vol. 14928, pp. 183–217. Springer, Cham (Aug 2024). https://doi.org/10.1007/978-3-031-68400-5_6

[39] Li, S., Wang, A., Zhang, C., Wang, X.: Two-server sublinear PIR with symmetric privacy and statistical security. Cryptology ePrint Archive, Report 2025/1313 (2025), https://eprint.iacr.org/2025/1313

[40] Liu, F.H., Wang, Z.: Rounding in the rings. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020, Part II. LNCS, vol. 12171, pp. 296–326. Springer, Cham (Aug 2020). https://doi.org/10.1007/978-3-030-56880-1$_1$1

[41] Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 1–23. Springer, Berlin, Heidelberg (May / Jun 2010). https://doi.org/10.1007/978-3-642-13190-5$_1$

[42] Mahdavi, R.A., Patel, S., Seo, J.Y., Yeo, K.: InsPIRe: Communication-efficient PIR with server-side preprocessing. Cryptology ePrint Archive, Paper 2025/1352 (2025), https://eprint.iacr.org/2025/1352

[43] Masny, D., Rindal, P.: Endemic oblivious transfer. In: Cavallaro, L., Kinder, J., Wang, X., Katz, J. (eds.) ACM CCS 2019. pp. 309–326. ACM Press (Nov 2019). https://doi.org/10.1145/3319535.3354210

[44] Menon, S.J., Wu, D.J.: YPIR: High-throughput single-server PIR with silent preprocessing. In: Proceedings of the 33rd USENIX Conference on Security Symposium. SEC '24, USENIX Association, USA (2024)

[45] Micciancio, D., Mol, P.: Pseudorandom knapsacks and the sample complexity of LWE search-to-decision reductions. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 465–484. Springer, Berlin, Heidelberg (Aug 2011). https://doi.org/10.1007/978-3-642-22792-9$_2$6

[46] Micciancio, D., Peikert, C.: Trapdoors for lattices: Simpler, tighter, faster, smaller. In: Pointcheval, D., Johansson, T. (eds.) Advances in Cryptology – EUROCRYPT 2012. pp. 700–718. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)

[47] Mughees, M.H., Pestana, G., Davidson, A., Livshits, B.: Privatefetch: Scalable catalog delivery in privacy-preserving advertising (2021), https://arxiv.org/abs/2109.08189

[48] Naor, M., Pinkas, B.: Oblivious transfer and polynomial evaluation. In: Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing. p. 245–254. STOC '99, Association for Computing Machinery, New York, NY, USA (1999). https://doi.org/10.1145/301250.301312

[49] Naor, M., Pinkas, B.: Oblivious transfer and polynomial evaluation. In: 31st ACM STOC. pp. 245–254. ACM Press (May 1999). https://doi.org/10.1145/301250.301312

[50] Patel, S., Persiano, G., Yeo, K.: Private stateful information retrieval. In: Lie, D., Mannan, M., Backes, M., Wang, X. (eds.) ACM CCS 2018. pp. 1002–1019. ACM Press (Oct 2018). https://doi.org/10.1145/3243734.3243821

[51] Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. J. ACM **56**(6) (Sep 2009). https://doi.org/10.1145/1568318.1568324

[52] Rossi, M.: Extended Security of Lattice-Based Cryptography. Theses, Équipe CASCADE, Département d'Informatique de l'ENS de Paris ; Université PSL (Sep 2020), https://hal.science/tel-02946399

[53] Servan-Schreiber, S., Hogan, K., Devadas, S.: AdVeil: A private targeted advertising ecosystem. Cryptology ePrint Archive, Paper 2021/1032 (2021), https://eprint.iacr.org/2021/1032

[54] Software, B.: Frodopir: a new privacy-preserving approach for retrieving data, https://brave.com/blog/frodopir/, https://brave.com/blog/frodopir/. 2022. Accessed 12th February 2026.

[55] of Standards, N.I., Technology: Sha-3 standard: Permutation-based hash and extendable-output functions. Tech. Rep. Federal Information Processing Standards Publications (FIPS) 202, August 05, 2015, U.S. Department of Commerce, Washington, D.C. (2001). https://doi.org/10.6028/nist.fips.202

[56] Vadhan, S.P.: Pseudorandomness. Foundations and Trends® in Theoretical Computer Science **7**(1–3), 1–336 (2012). https://doi.org/10.1561/0400000010

[57] Wu, Z., Zhang, D., Li, Y., Han, X.: ASKPIR: Authorized symmetric keyword privacy information retrieval protocol based on DID. Cryptology ePrint Archive, Paper 2023/1631 (2023), https://eprint.iacr.org/2023/1631

[58] Zhou, M., Park, A., Zheng, W., Shi, E.: Piano: Extremely simple, single-server pir with sublinear server computation. In: 2024 IEEE Symposium on Security and Privacy (SP). pp. 4296–4314 (2024). https://doi.org/10.1109/SP54263.2024.00055