Barely Doubly-Efficient PIR from Quadratic Residuosity

Keewoo Lee

Ethereum Foundation
keewoo.lee@ethereum.org

Abstract. A Private Information Retrieval (PIR) scheme allows a client to retrieve data from a database hosted on a remote server without revealing which location is being accessed. In Doubly-Efficient PIR (DEPIR), the server preprocesses the database offline into a data structure that enables it to answer any client query in sublinear time with respect to the database size n. Lin, Mook, and Wichs (STOC 2023) presented the first DEPIR, from RLWE, and Lee (Crypto 2026) gave a barely doubly-efficient scheme from LWE in the CRS model, answering each query in $o(n/\log n)$ time. These are essentially the only constructions known, and both rest on lattices. Whether DEPIR could be realized from non-lattice-based foundations, particularly from number-theoretic assumptions, remained in doubt.

In this work, we construct a DEPIR scheme from quadratic residuosity in the CRS model, answering each query in $o(n/\log n)$ time. As in Lee's LWE-based construction, our scheme leverages Williams's matrix-vector multiplication preprocessing (SODA 2007). Privacy follows from the pseudorandomness of Legendre symbols under quadratic residuosity, proved by Corrigan-Gibbs and Wu (TCC 2025). To cope with a new modulus at every query, we additionally develop a data structure that reduces a preprocessed integer modulo any modulus in sublinear time.

1 Introduction

In (single-server) Private Information Retrieval (PIR) [CGKS95, KO97], a client wishes to retrieve an entry from a remote database maintained by an untrusted server, while ensuring that the query index remains private. A classic lower bound for PIR, due to Beimel, Ishai, and Malkin [BIM00], states that the server must perform at least $\Omega(n)$ computation, where n is the size of the database. Otherwise, some entries of the database would be left untouched during computation, thereby compromising query privacy. This lower bound, however, assumes that the server accesses the database *as-is*.

In the same paper [BIM00], they demonstrated that sublinear server-side computation is possible if *preprocessing* is allowed. However, their construction is in the non-colluding multi-server setting, and they left open the question of whether such efficiency could be achieved in the single-server setting

with preprocessing. This primitive was later termed Doubly-Efficient PIR (DEPIR) [CHR17].[1] The problem remained open for over two decades and was often regarded as too good to be true. Notably, [BIPW17] even devotes part of its discussion to explaining why proving the impossibility of DEPIR seems unlikely.

This changed with the breakthrough of Lin, Mook, and Wichs [LMW23], which presented the first construction of DEPIR, based on the RLWE assumption [LPR10], achieving $\mathsf{polylog}(n)$ online runtime. Follow-up work has since improved its concrete efficiency [DMZ23,OPPW24,OPPW25,XZO+26], though the construction remains far from practical. Lee [Lee26] instead explored a weaker assumption, giving a construction from LWE [Reg05] in the Common Reference String (CRS) model, only *barely* doubly-efficient, answering each query in $o(n/\log n)$ time. Note that all of these constructions are lattice-based. Prior to our work, whether DEPIR is the exclusive province of lattices, or can also be based on number-theoretic assumptions such as quadratic residuosity [GM82], remained in doubt.

1.1 Our Contribution

In this work, we present a DEPIR construction based on quadratic residuosity in the CRS model (Figure 1 and Theorem 4.3). For an arbitrarily small constant $\varepsilon > 0$, its preprocessing takes $n^{1+\varepsilon} \cdot \mathsf{poly}(\lambda)$ time. The server then answers each query in $O\big(n/(\varepsilon \log n)^2\big)$ time, provided $n \geq \lambda^c$ for a sufficiently large constant c, so the scheme is only barely doubly-efficient, as is the construction of Lee [Lee26]. For instance, one may take $\varepsilon = 1/4$ and $c = 3$.

We do not expect the scheme to be useful in practice, but its existence answers a question about which assumptions can support DEPIR. Lin, Mook, and Wichs [LMW25] show that the black-box use of essentially any cryptographic primitive, idealized ones included, is useless for DEPIR, so a construction has no choice but to lean on the structure of a concrete assumption. Thus the question becomes which assumptions carry enough structure. RLWE suffices [LMW23], and LWE suffices in the CRS model [Lee26]. We show that quadratic residuosity suffices as well, in the CRS model.

This can be read in two directions. One chips away at the unique position of lattice assumptions, which nothing else matches for tasks like fully homomorphic encryption. DEPIR can be obtained without lattices, at least once a CRS is allowed. The other is that quadratic residuosity carries more than one might think. Neither the decisional Diffie-Hellman nor the RSA assumption is known to support DEPIR, which sets quadratic residuosity apart from them. It would be interesting to narrow that gap, by building DEPIR from those assumptions, or to widen it, by drawing more out of quadratic residuosity.

[1] We use the term DEPIR in its narrow, unkeyed sense (Definition 2.1), where the preprocessed database is public and anyone can query it. Keyed variants [BIPW17, CHR17, CIMR25, CIJ+26], whose preprocessing is performed with a key, are an exciting line of work in their own right, but lie outside the scope of this paper.

1.2 Technical Overview

At a high level, our scheme is the construction of Lee [Lee26] with the Legendre symbol in place of LWE samples. We therefore begin by reviewing it. The descriptions here are deliberately imprecise, so that the ideas are not buried in details. The precise versions are in [Lee26] and in Figure 1.

Review: LWE-based DEPIR. Lee's LWE-based construction is essentially SimplePIR [HHC+23] with its matrix-vector multiplication preprocessed by the method of Williams [Wil07]. The database is arranged as a $\sqrt{n} \times \sqrt{n}$ matrix $\boldsymbol{D}$, and the CRS is a public matrix $\boldsymbol{A}$. A client that wants column $\hat{\jmath}$ draws a secret $\boldsymbol{s}$ and sets $\boldsymbol{b} = \boldsymbol{A}\boldsymbol{s} + \boldsymbol{e}$ with $\boldsymbol{e}$ small, so that $(\boldsymbol{A}, \boldsymbol{b})$ is an LWE sample. Given $\boldsymbol{A}$, such a $\boldsymbol{b}$ is pseudorandom. The client masks the unit vector with the sample, sending $\boldsymbol{v} = \boldsymbol{b} + \boldsymbol{u}_{\hat{\jmath}}$, where $\boldsymbol{u}_{\hat{\jmath}}$ has a one in position $\hat{\jmath}$ and zeros elsewhere. After preprocessing $\boldsymbol{D}$, the server can return $\boldsymbol{w} = \boldsymbol{D}\boldsymbol{v}$ in $O(n/\log^2 n)$ time, together with the hint $\boldsymbol{H} = \boldsymbol{D}\boldsymbol{A}$. As $\boldsymbol{w} = \boldsymbol{D}\boldsymbol{b} + \boldsymbol{D}\boldsymbol{u}_{\hat{\jmath}}$, the client can unmask $\boldsymbol{w}$ with $\boldsymbol{D}\boldsymbol{b} = \boldsymbol{D}(\boldsymbol{A}\boldsymbol{s} + \boldsymbol{e}) = \boldsymbol{H}\boldsymbol{s} + \boldsymbol{D}\boldsymbol{e}$, which it reconstructs from $\boldsymbol{H}$ and its own $\boldsymbol{s}$ up to the small term $\boldsymbol{D}\boldsymbol{e}$. The hint depends only on the CRS and the database, so it is the same for every query and can be computed once during preprocessing.

Starting Point: Kushilevitz-Ostrovsky Scheme. Our construction starts from the very first single-server PIR scheme, due to Kushilevitz and Ostrovsky [KO97], which rests on the encryption scheme of Goldwasser and Micali [GM82]. Its query encrypts $\boldsymbol{u}_{\hat{\jmath}}$ under a modulus $N = pq$ that the client samples. That is, it is a vector $\boldsymbol{x}$ of integers with $L_p(\boldsymbol{x}) = \boldsymbol{u}_{\hat{\jmath}}$, where the Legendre bit $L_p(x) \in \mathbb{Z}_2$ is zero when x is a square modulo p and one when it is not. The client keeps p and q to itself, and under quadratic residuosity $\boldsymbol{x}$ hides $\hat{\jmath}$ from anyone without them. The server returns the row products $\prod_j x_j^{\boldsymbol{D}[i,j]}$, one for each row i, whose Legendre bits are $\boldsymbol{D}\boldsymbol{u}_{\hat{\jmath}}$ by multiplicativity, and the client reads off the entry it asked for, using p. Here the database sits in the exponents, and the bases x_j arrive with the query. We know of no preprocessing for a product of that kind, and as it stands the server performs up to n modular multiplications for every query.

Main Idea: Masking with Legendre Bits. To bring the scheme closer to Lee's, we *mask* the unit vector rather than encode it in the integers $\boldsymbol{x}$. Draw $\boldsymbol{a}$ uniformly and publish it as a CRS, and let the client send N together with $\boldsymbol{v} = L_p(\boldsymbol{a}) + \boldsymbol{u}_{\hat{\jmath}}$, a vector of length $\sqrt{n}$ over $\mathbb{Z}_2$. The pair $(\boldsymbol{a}, L_p(\boldsymbol{a}))$ now plays the role of $(\boldsymbol{A}, \boldsymbol{A}\boldsymbol{s}+\boldsymbol{e})$, with the prime p in the role of the secret $\boldsymbol{s}$. Corrigan-Gibbs and Wu [CW25] show that the mask $L_p(\boldsymbol{a})$ is pseudorandom given $\boldsymbol{a}$ and N, so the query hides the column (less immediate than it looks, as the CRS is fixed before the modulus, and Lemma 4.2 gives the argument). The Legendre symbol is also a homomorphism, so $\boldsymbol{D}L_p(\boldsymbol{a}) = L_p(\boldsymbol{h})$ for the vector $\boldsymbol{h}$ of row products $h_i = \prod_j a_j^{\boldsymbol{D}[i,j]}$ (over $\mathbb{Z}$), which can be formed before any client appears and plays the role of Lee's hint $\boldsymbol{H}$. The server sends $\boldsymbol{h}$ with the answer $\boldsymbol{w} = \boldsymbol{D}\boldsymbol{v}$, which the preprocessing of Williams gives in $O(n/\log^2 n)$ time. As $\boldsymbol{w} = \boldsymbol{D}L_p(\boldsymbol{a}) + \boldsymbol{D}\boldsymbol{u}_{\hat{\jmath}}$, the client can unmask $\boldsymbol{w}$ with $L_p(\boldsymbol{h})$, which it computes from the $\boldsymbol{h}$ it receives and its own p.

Challenge: A Fresh Modulus for Every Query. In SimplePIR and Lee's construction, the client must draw a fresh secret $\boldsymbol{s}$ for every query, and ours fresh primes p and q. The difference is that our secret decides the modulus, so the residues $\boldsymbol{h} \bmod N$ that travel with the answer are different every time, where Lee's hint is the same every time. Sending $\boldsymbol{h}$ over $\mathbb{Z}$ is not an option, as its entries are about $\sqrt{n}$ times as long as the modulus and together outweigh the database, and reducing them from scratch at every query would cost as much. In Section 3, we therefore develop a tool that preprocesses an integer so that its residue modulo a modulus supplied online can be computed in sublinear time, in the spirit of the fast evaluation of Kedlaya and Umans [KU08] that drives the construction of Lin, Mook, and Wichs [LMW23]. It may be of independent interest.

2 Preliminaries

2.1 Notations

We use $[n]$ to denote $\{0, 1, \ldots, n-1\}$, and vectors and matrices are written in bold, with entries $\boldsymbol{v}[i]$ and $\boldsymbol{D}[i, j]$ indexed from 0. *Negligible* and *overwhelming* are always with respect to the security parameter λ, and we write $\approx_c$ for computational indistinguishability of distributions and $\equiv$ for equality of distributions.

2.2 Doubly-Efficient Private Information Retrieval

Definition 2.1 (DEPIR). *A* Doubly-Efficient Private Information Retrieval (DEPIR) *scheme consists of a tuple of probabilistic algorithms* $(\mathtt{Setup}, \mathtt{Prep}, \mathtt{Query}, \mathtt{Answer}, \mathtt{Extract})$*, each described below.*

- $\mathtt{pp} \leftarrow \mathtt{Setup}(1^\lambda, 1^n)$*: The setup algorithm takes as input the security parameter* λ *and the database size* n*, and outputs the public parameter* $\mathtt{pp}$*. All subsequent algorithms take* $\mathtt{pp}$ *as an implicit input.*
- $\widetilde{\mathtt{DB}} \leftarrow \mathtt{Prep}(\mathtt{DB})$*: The preprocessing algorithm takes as input a database* $\mathtt{DB} \in \{0,1\}^n$ *and outputs a preprocessed database* $\widetilde{\mathtt{DB}}$*.*
- $(\mathtt{q}, \mathtt{st}) \leftarrow \mathtt{Query}(i)$*: The query algorithm takes as input a query index* $i \in [n]$ *and outputs a query* $\mathtt{q}$ *along with a secret state* $\mathtt{st}$*.*
- $\mathtt{a} \leftarrow \mathtt{Answer}(\widetilde{\mathtt{DB}}, \mathtt{q})$*: The answer algorithm takes as input a preprocessed database* $\widetilde{\mathtt{DB}}$ *(stored in random-access memory) and a query* $\mathtt{q}$*, and outputs an answer* $\mathtt{a}$*.*
- $\mathtt{b} \leftarrow \mathtt{Extract}(\mathtt{st}, \mathtt{a})$*: The extract algorithm takes a secret state* $\mathtt{st}$ *and an answer* $\mathtt{a}$ *as input, and outputs a bit* $\mathtt{b} \in \{0,1\}$*.*

The algorithms should satisfy the following correctness, privacy, and efficiency.

***Correctness*:** *For any* $\mathtt{DB} \in \{0,1\}^n$ *and any* $i \in [n]$*, the following probability is overwhelming for an overwhelming fraction of* $\mathtt{pp} \leftarrow \mathtt{Setup}(1^\lambda, 1^n)$*.*

$$\Pr\left[\begin{array}{l|r} \mathtt{Extract}(\mathtt{st}, \mathtt{a}) = \mathtt{DB}[i] & \widetilde{\mathtt{DB}} \leftarrow \mathtt{Prep}(\mathtt{DB}) \\ & (\mathtt{q}, \mathtt{st}) \leftarrow \mathtt{Query}(i) \\ & \mathtt{a} \leftarrow \mathtt{Answer}(\widetilde{\mathtt{DB}}, \mathtt{q}) \end{array}\right]$$

***Privacy*:** *For any* $i_0, i_1 \in [n]$*, no probabilistic polynomial-time adversary given* $\mathtt{pp}$ *can distinguish the query* $\mathtt{q}$ *of* $\mathtt{Query}(i_0)$ *from that of* $\mathtt{Query}(i_1)$ *with non-negligible advantage, for an overwhelming fraction of* $\mathtt{pp} \leftarrow \mathtt{Setup}(1^\lambda, 1^n)$*.*

***Efficiency*:** *The* $\mathtt{Setup}$ *and* $\mathtt{Prep}$ *algorithms run in* $\mathsf{poly}(\lambda, n)$ *time, while the* $\mathtt{Query}$*,* $\mathtt{Answer}$*, and* $\mathtt{Extract}$ *algorithms each run in* $o(n) \cdot \mathsf{poly}(\lambda)$ *time.*

2.3 Quadratic Residuosity

Definition 2.2 (Legendre Symbol). *For an odd prime* p *and an integer* a *with* $p \nmid a$*, the Legendre symbol* $\left(\frac{a}{p}\right)$ *is* $+1$ *if* a *is a square modulo* p *and* -1 *otherwise. Let* $L_p(a) \in \mathbb{Z}_2$ *be the bit with*

$$\left(\frac{a}{p}\right) = (-1)^{L_p(a)},$$

and let $L_p(a) = 0$ *if* $p \mid a$*. By the multiplicativity of the Legendre symbol,* $L_p(ab) = L_p(a) + L_p(b)$ *whenever* $p \nmid ab$*. We apply* L_p *coordinatewise to vectors, giving* $L_p(\boldsymbol{a}) \in \mathbb{Z}_2^m$ *for* $\boldsymbol{a} \in \mathbb{Z}^m$*.*

Definition 2.3 (Quadratic Residuosity). *A* Blum prime *is a prime* p *satisfying* $p = 3 \pmod 4$*, and* $\mathtt{Gen}(1^\lambda)$ *samples an ordered pair of distinct uniformly random* λ*-bit Blum primes. Let*

$$(p, q) \leftarrow \mathtt{Gen}(1^\lambda), \qquad N = pq < 2^{2\lambda}.$$

The quadratic-residuosity (QR) assumption *states that, for* $x \leftarrow \mathbb{Z}_N^\times$*,*

$$(N, x^2) \approx_c (N, -x^2).$$

Lemma 2.4 ([CW25, Cor. 3.6]). *Under the QR assumption (Definition 2.3), for any* $m = \mathsf{poly}(\lambda)$*,*

$$(\boldsymbol{x}, N, L_p(\boldsymbol{x})) \approx_c (\boldsymbol{x}, N, \boldsymbol{r}),$$

where $(p, q) \leftarrow \mathtt{Gen}(1^\lambda)$*,* $N = pq$*,* $\boldsymbol{x} \leftarrow (\mathbb{Z}_N^\times)^m$*, and* $\boldsymbol{r} \leftarrow \mathbb{Z}_2^m$ *are sampled independently. A distinguisher with advantage* δ *yields a QR distinguisher with advantage at least* $\delta/(2m)$*.*

2.4 Fast Matrix-Vector Multiplication with Preprocessing

Lemma 2.5 ([Wil07]). *For all* $\varepsilon \in (0, 1)$*, every* $m \times m$ *matrix over* $\mathbb{Z}_2$ *can be preprocessed in* $O(m^{2+\varepsilon})$ *time so that every subsequent matrix-vector multiplication can be performed in* $O(m^2/(\varepsilon \log m)^2)$ *steps on an* $O(\log m)$*-word RAM.*

3 Fast Modular Reduction with Preprocessing

In this section, we develop a data structure that preprocesses an integer so that its residue modulo a modulus supplied online can be computed in sublinear time (Theorem 3.2). Following Kedlaya and Umans [KU08], we write the integer as a multivariate polynomial of low individual degree, tabulate it modulo small primes offline, and evaluate it online by table lookups and the Chinese Remainder Theorem (CRT). The following lemma says which primes suffice.

Lemma 3.1 ([KU08, Lem. 2.3]). *For all integers $M \geq 2$, the product of the primes less than $16 \log M$ is greater than M.*

Theorem 3.2. *For every constant $\varepsilon \in (0, 1/2)$ and all integers $\gamma \geq \beta \geq 2$, every integer $a \in [2^\gamma]$ can be preprocessed so that afterwards, for every modulus $2 \leq N < 2^\beta$, the residue $a \bmod N$ can be computed, with the following performance on an $O(\log \gamma)$-word RAM.*

- *Preprocessing Time:* $(1/\varepsilon)^{O(1/\varepsilon)} \cdot \widetilde{O}\big(\beta^{1+1/\varepsilon} \cdot (\gamma/\beta)^{1+\varepsilon}\big)$
- *Online Time:* $\widetilde{O}\big(\beta \cdot (\gamma/\beta)^{\varepsilon}/\varepsilon^2\big)$

Proof. We first identify $a \in [2^\gamma]$ with a multivariate polynomial F_a over $\mathbb{Z}$. Let $d, k, b \geq 1$ be integers with $d^k b \geq \gamma$, to be fixed below, and let $B = 2^b$. Then $a < 2^\gamma \leq B^{d^k}$. For each $n \in [d^k]$, let $a_n \in [B]$ be the n-th digit of a in base B, so that $a = \sum_{n \in [d^k]} a_n B^n$. Likewise, for each $n \in [d^k]$ and $i \in [k]$, let $n_i \in [d]$ be the i-th digit of n in base d, so that $n = \sum_{i \in [k]} n_i d^i$. Consider the k-variate polynomial

$$F_a(X_0, \ldots, X_{k-1}) = \sum_{n \in [d^k]} a_n \prod_{i \in [k]} X_i^{n_i} \in \mathbb{Z}[X_0, \ldots, X_{k-1}].$$

By construction, its coefficients lie in $[B]$, its individual degrees are less than d, and $a = F_a(B, B^d, \ldots, B^{d^{k-1}})$. Hence, for every modulus $N < 2^\beta$ and the point $\boldsymbol{x} = (x_0, \ldots, x_{k-1})$ with $x_i \in [N]$ such that $x_i = B^{d^i} \bmod N$, we have

$$F_a(\boldsymbol{x}) = a \pmod N \qquad \text{and} \qquad 0 \leq F_a(\boldsymbol{x}) < M, \tag{1}$$

where $M := d^k \cdot 2^{b+\beta k d}$, as each of the d^k terms of $F_a(\boldsymbol{x})$ is less than $2^b \cdot 2^{\beta k d}$ by $x_i < N < 2^\beta$. Note that F_a does not depend on N, and $\boldsymbol{x}$ does not depend on a.

Preprocessing. To preprocess $a \in [2^\gamma]$, we build lookup tables for F_a modulo small primes. By Lemma 3.1, the product of the primes less than $16 \log M$ exceeds M. Let $q_0 < \cdots < q_{t-1}$ be these primes, so that $t < 16 \log M$. For every $j \in [t]$, store the complete table $T_j \colon [q_j]^k \to [q_j]$ of F_a modulo q_j. To fill T_j, reduce the d^k coefficients of F_a modulo q_j and substitute the variables one at a time. For each $i \in [k]$ in turn, substituting every value in $[q_j]$ for X_i turns q_j^i polynomials with d^{k-i} coefficients into q_j^{i+1} polynomials with d^{k-i-1} coefficients, using $O(q_j^{i+1} \cdot d^{k-i})$ operations modulo q_j. As $q_j < 16 \log M$ and

$d \le \log M$, this is $2^{O(k)} \cdot d \cdot (\log M)^k$ operations. Charging $\mathsf{polylog}(\log M)$ per operation and summing over the k substitutions and the $t < 16 \log M$ primes, the preprocessing time is

$$2^{O(k)} \cdot \widetilde{O}\big(d \cdot (\log M)^{k+1}\big). \tag{2}$$

The coefficient reductions take $\widetilde{O}(d^k \cdot b)$ time per table, as the d^k coefficients have b bits, and, summed over the t tables, are within (2), as $t < 16 \log M$ and $d^{k-1} b \le (\log M)^k$.

Online evaluation. Given N, compute $x_0 = B \bmod N$ and $x_{i+1} = x_i^d \bmod N$ for $i \in [k-1]$ by $O(k \log(bd))$ multiplications modulo N, in $\widetilde{O}(k\beta \log\log M)$ time with fast integer multiplication, as $b, d \le \log M$. Reduce each x_i modulo all q_j at once by a remainder tree [vzGG13, Sec. 10.3], in $\widetilde{O}(\log M)$ time, and look up $T_j(x_0 \bmod q_j, \ldots, x_{k-1} \bmod q_j) = F_a(\boldsymbol{x}) \pmod{q_j}$ for every $j \in [t]$. By (1), fast CRT [vzGG13, Sec. 10.3] recovers $F_a(\boldsymbol{x})$ from these residues, and the output $F_a(\boldsymbol{x}) \bmod N$ equals $a \bmod N$, in $\widetilde{O}(\log M)$ time as well. The t lookups, each addressed by k residues, take $O(k \log M)$ time as long as a residue fits in a word, that is, $\log\log M = O(\log \gamma)$, which our parameters below satisfy. As $\beta \le \log M$, the total online time is

$$\widetilde{O}(k \log M). \tag{3}$$

Parameters. Now let $b = \beta d$ and $d = \lceil (\gamma/\beta)^{1/(k+1)} \rceil$, so that $d^k b \ge \gamma$. Let $k = \lfloor 1/\varepsilon \rfloor$, so that $2 \le k \le 1/\varepsilon$ and $1/(k+1) < \varepsilon$. As $\gamma \ge \beta$, we have $d \le 2(\gamma/\beta)^{1/(k+1)} \le 2(\gamma/\beta)^{\varepsilon}$, and in particular $d^{k+1} \le 2^{k+1}\gamma/\beta$. Since each of the three terms of $\log M = k \log d + b + \beta k d$ is at most $\beta k d$, we have $\log M \le 3\beta k d$. Hence the online time (3) is at most

$$\widetilde{O}\big(k^2 \cdot \beta \cdot d\big) \le \widetilde{O}\big(\beta \cdot (\gamma/\beta)^{\varepsilon}/\varepsilon^2\big),$$

using $d \le 2(\gamma/\beta)^{\varepsilon}$ and $k \le 1/\varepsilon$, and, by $\log M \le 3\beta k d$ and $k \ge 2$, the preprocessing time (2) is at most

$$k^{O(k)} \cdot \widetilde{O}\big(\beta^{k+1} \cdot d^{k+2}\big) \le (1/\varepsilon)^{O(1/\varepsilon)} \cdot \widetilde{O}\big(\beta^{1+1/\varepsilon} \cdot (\gamma/\beta)^{1+\varepsilon}\big).$$

Here the inequality uses $d^{k+2} = d^{k+1} \cdot d \le 2^{k+2} \cdot (\gamma/\beta)^{1+\varepsilon}$, where 2^{k+2} is absorbed into $k^{O(k)}$, and then $k^{O(k)} \le (1/\varepsilon)^{O(1/\varepsilon)}$ and $\beta^{k+1} \le \beta^{1+1/\varepsilon}$, as $k \le 1/\varepsilon$. Finally, the parameters satisfy $\log\log M = O(\log \gamma)$, as used in the online evaluation: $d \le 2(\gamma/\beta)^{\varepsilon} \le 2\gamma/\beta$ gives $\log M \le 3\beta k d \le 6k\gamma$, and $k \le 1/\varepsilon$ is a constant. □

4 QR-Based DEPIR in the CRS Model

In this section, we present our DEPIR scheme from quadratic residuosity in the CRS model (Figure 1). We prove correctness (Lemma 4.1), privacy (Lemma 4.2), and efficiency (Theorem 4.3). An informal overview of the design is in Section 1.2.

QR-Based DEPIR in the CRS Model

$\mathtt{pp} \leftarrow \mathtt{Setup}(1^\lambda, 1^n)$:

1. Set $m = \lceil \sqrt{n} \rceil$ and $\beta = 3\lambda$.
2. Sample $\boldsymbol{a} = (a_0, \ldots, a_{m-1}) \leftarrow ([2^\beta] \setminus \{0\})^m$ as CRS.
3. Output $\mathtt{pp} = (n, m, \beta, \boldsymbol{a})$.

$\widetilde{\mathtt{DB}} \leftarrow \mathtt{Prep}(\mathtt{DB})$:

1. Reshape $\mathtt{DB} \in \{0,1\}^n$ into $\boldsymbol{D} \in \mathbb{Z}_2^{m \times m}$ with $\boldsymbol{D}[i,j] = \mathtt{DB}[i + j \cdot m]$ for $i, j \in [m]$.
2. Preprocess $\boldsymbol{D}$ into $\widetilde{\boldsymbol{D}}$ using Lemma 2.5.
3. For $i \in [m]$, compute the integer $h_i \leftarrow \prod_{j \in [m]} a_j^{\boldsymbol{D}[i,j]}$, which has at most $m\beta$ bits, and preprocess it into $\widetilde{h}_i$ using Theorem 3.2, and let $\widetilde{\boldsymbol{h}} = (\widetilde{h}_0, \ldots, \widetilde{h}_{m-1})$.
4. Output $\widetilde{\mathtt{DB}} = (\widetilde{\boldsymbol{D}}, \widetilde{\boldsymbol{h}})$.

$(\mathtt{q}, \mathtt{st}) \leftarrow \mathtt{Query}(i)$:

1. Compute $\hat{\imath}, \hat{\jmath} \in [m]$ such that $i = \hat{\imath} + \hat{\jmath} \cdot m$.
2. Sample $(p, q) \leftarrow \mathtt{Gen}(1^\lambda)$ (Definition 2.3) and set $N \leftarrow pq$.
3. If $\gcd(a_j, N) > 1$ for some $j \in [m]$, output $\mathtt{q} = \perp$ and $\mathtt{st} = \perp$.
4. Compute $\boldsymbol{v} \leftarrow L_p(\boldsymbol{a}) + \boldsymbol{u}_{\hat{\jmath}}$ (Definition 2.2), where $\boldsymbol{u}_{\hat{\jmath}} \in \mathbb{Z}_2^m$ denotes the unit vector with a one in the $\hat{\jmath}$-th position and zeros elsewhere.
5. Output $\mathtt{q} = (\boldsymbol{v}, N)$ and $\mathtt{st} = (p, \hat{\imath})$.

$\mathtt{a} \leftarrow \mathtt{Answer}(\widetilde{\mathtt{DB}}, \mathtt{q})$:

1. If $\mathtt{q} = \perp$, output $\mathtt{a} = \perp$.
2. Parse $\widetilde{\mathtt{DB}}$ as $(\widetilde{\boldsymbol{D}}, \widetilde{\boldsymbol{h}})$ and $\mathtt{q}$ as $(\boldsymbol{v}, N)$.
3. Evaluate $\boldsymbol{w} \leftarrow \boldsymbol{D}\boldsymbol{v}$ with $\widetilde{\boldsymbol{D}}$ using Lemma 2.5.
4. For $i \in [m]$, compute $\check{h}_i \leftarrow h_i \bmod N$ with $\widetilde{h}_i$ using Theorem 3.2, and let $\check{\boldsymbol{h}} = (\check{h}_0, \ldots, \check{h}_{m-1})$.
5. Output $\mathtt{a} = (\boldsymbol{w}, \check{\boldsymbol{h}})$.

$\mathtt{b} \leftarrow \mathtt{Extract}(\mathtt{st}, \mathtt{a})$:

1. If $\mathtt{st} = \perp$, output $\mathtt{b} = 0$.
2. Parse $\mathtt{st}$ as $(p, \hat{\imath})$ and $\mathtt{a}$ as $(\boldsymbol{w}, \check{\boldsymbol{h}})$.
3. Output $\mathtt{b} = \boldsymbol{w}[\hat{\imath}] + L_p(\check{h}_{\hat{\imath}})$ (Definition 2.2).

Fig. 1. A description of our QR-based DEPIR in the CRS model. The public parameter $\mathtt{pp}$ is assumed to be an implicit input to all algorithms other than $\mathtt{Setup}$.

Lemma 4.1 (Correctness). *The scheme of Figure 1 is correct in the sense of Definition 2.1. Specifically,* `Query` *aborts with probability* $O(m\lambda \cdot 2^{-\lambda})$.

Proof. On a non-abort query every a_j is a unit modulo N, so every h_i is a unit modulo N and $L_p(\check{\boldsymbol{h}}) = \boldsymbol{D}L_p(\boldsymbol{a})$ by construction. Hence

$$\boldsymbol{w} + L_p(\check{\boldsymbol{h}}) = \boldsymbol{D}\big(L_p(\boldsymbol{a}) + \boldsymbol{u}_{\hat{\jmath}}\big) + \boldsymbol{D}L_p(\boldsymbol{a}) = \boldsymbol{D}\boldsymbol{u}_{\hat{\jmath}},$$

so $\boldsymbol{w}[\hat{\imath}] + L_p(\check{h}_{\hat{\imath}}) = \boldsymbol{D}[\hat{\imath}, \hat{\jmath}] = \mathtt{DB}[i]$. For the abort probability, fix $\boldsymbol{a}$. A positive integer below 2^β has at most $\beta/(\lambda-1) = O(1)$ distinct λ-bit prime factors, so $O(m)$ distinct λ-bit primes divide some a_j. There are $\Theta(2^\lambda/\lambda)$ λ-bit Blum primes by the prime number theorem for arithmetic progressions, so the probability that p or q is one of these divisors is $O(m\lambda \cdot 2^{-\lambda})$. □

Lemma 4.2 (Privacy). *Under the QR assumption (Definition 2.3), the scheme of Figure 1 is private in the sense of Definition 2.1. A distinguisher between the queries for two indices with advantage* δ*, averaged over* `pp`*, yields a QR distinguisher with advantage at least* $(\delta - O(m \cdot 2^{-\lambda}))/(4m)$.

Proof. We first show that the mask $L_p(\boldsymbol{a})$ is pseudorandom given the CRS $\boldsymbol{a}$ and the modulus N, and then derive query privacy from it.

Mask pseudorandomness. Let $\boldsymbol{a}$ be the CRS and let $N = pq$ be the modulus of a query, sampled independently of $\boldsymbol{a}$. We show that

$$(\boldsymbol{a}, N, L_p(\boldsymbol{a})) \approx_c (\boldsymbol{a}, N, \boldsymbol{r}), \qquad \text{where } \boldsymbol{r} \leftarrow \mathbb{Z}_2^m. \tag{4}$$

Let $\boldsymbol{a}' = (a'_0, \ldots, a'_{m-1})$ be sampled by drawing $\boldsymbol{x} = (x_0, \ldots, x_{m-1}) \leftarrow (\mathbb{Z}_N^\times)^m$ and, for every $j \in [m]$, a uniformly random lift $a'_j \in [2^\beta]$ with $a'_j = x_j \pmod N$, which needs only N and x_j. Every residue class modulo N has either $\lfloor 2^\beta/N \rfloor$ or $\lceil 2^\beta/N \rceil$ representatives in $[2^\beta]$, so a uniform nonzero element of $[2^\beta]$ and a uniform lift of a uniform element of $\mathbb{Z}_N^\times$ are within statistical distance $O(1/p + 1/q + N/2^\beta)$, the first two terms accounting for the excluded nonunit residues and the last for the unequal numbers of lifts and the excluded zero. As p and q are λ-bit, so that $N < 2^{2\lambda}$, and $\beta = 3\lambda$, this is $O(2^{-\lambda})$, so by a union bound over the m coordinates $(\boldsymbol{a}, N, p)$ and $(\boldsymbol{a}', N, p)$ are within statistical distance $O(m \cdot 2^{-\lambda})$. As both sides of (4) are computed from $(\boldsymbol{a}, N, p)$ and independent coins, it suffices to prove (4) with $\boldsymbol{a}'$ in place of $\boldsymbol{a}$, at the cost of $O(m \cdot 2^{-\lambda})$ in advantage.

With $\boldsymbol{a}'$, the two sides of (4) are the two distributions of Lemma 2.4 under the lifting $\boldsymbol{x} \mapsto \boldsymbol{a}'$: it maps $(\boldsymbol{x}, N, L_p(\boldsymbol{x}))$ to $(\boldsymbol{a}', N, L_p(\boldsymbol{a}'))$, since $L_p(\boldsymbol{a}') = L_p(\boldsymbol{x})$ as p divides N, and $(\boldsymbol{x}, N, \boldsymbol{r})$ to $(\boldsymbol{a}', N, \boldsymbol{r})$. Hence (4) holds, and a distinguisher for it with advantage δ yields a QR distinguisher with advantage at least $(\delta - O(m \cdot 2^{-\lambda}))/(2m)$ by Lemma 2.4.

Query privacy. Fix $i_0, i_1 \in [n]$ and let $\hat{\jmath}_0, \hat{\jmath}_1$ be their column positions. For $j \in [m]$, let $V_j(\boldsymbol{a}, N, \boldsymbol{z})$ be the server's view of a query with mask $\boldsymbol{z}$ and column j, a deterministic map. Then, for $b \in \{0, 1\}$, $V_{\hat{\jmath}_b}(\boldsymbol{a}, N, L_p(\boldsymbol{a}))$ is the server's view

of $\texttt{Query}(i_b)$, and the distribution of $V_{\hat{j}_b}(\boldsymbol{a}, N, \boldsymbol{r})$ does not depend on b, as the abort condition is decided by $(\boldsymbol{a}, N)$ alone and $\boldsymbol{r} + \boldsymbol{u}_{\hat{j}_b}$ is uniform. Hence

$$V_{\hat{j}_0}(\boldsymbol{a}, N, L_p(\boldsymbol{a})) \approx_c V_{\hat{j}_0}(\boldsymbol{a}, N, \boldsymbol{r}) \equiv V_{\hat{j}_1}(\boldsymbol{a}, N, \boldsymbol{r}) \approx_c V_{\hat{j}_1}(\boldsymbol{a}, N, L_p(\boldsymbol{a}))$$

by (4), so a distinguisher between $(\texttt{pp}, \texttt{Query}(i_0))$ and $(\texttt{pp}, \texttt{Query}(i_1))$ with advantage δ averaged over $\texttt{pp}$ distinguishes (4) with advantage at least $\delta/2$, and by mask pseudorandomness yields a QR distinguisher with advantage at least $(\delta - O(m \cdot 2^{-\lambda}))/(4m)$. Privacy for an overwhelming fraction of $\texttt{pp}$ follows by Markov's inequality. □

Theorem 4.3 (Main Theorem). *Under the QR assumption (Definition 2.3), for any constant $\varepsilon \in (0, 1/2)$, there exists a DEPIR scheme (Definition 2.1) in the CRS model with the following efficiency on an $O(\log n + \log \lambda)$-word RAM, where n is the database size, λ is the security parameter, and $T_{\texttt{Gen}}(\lambda)$ is the cost of $\texttt{Gen}(1^\lambda)$ (Definition 2.3).*

- *Preprocessing Runtime:* $(1/\varepsilon)^{O(1/\varepsilon)} \cdot \widetilde{O}\big(\lambda^{1+1/\varepsilon} \cdot n^{1+\varepsilon/2}\big)$
- *Server's Online Runtime:* $O\big(n/(\varepsilon \log n)^2\big) + \widetilde{O}\big(\lambda \cdot n^{1/2+\varepsilon/2}/\varepsilon^2\big)$
- *Client's Online Runtime:* $T_{\texttt{Gen}}(\lambda) + \widetilde{O}\big(\sqrt{n} \cdot \lambda\big)$
- *Online Communication:* $O(\sqrt{n} \cdot \lambda)$

In particular, if $n \geq \lambda^c$ for a constant $c > 2/(1-\varepsilon)$, then the server's online runtime is $O\big(n/(\varepsilon \log n)^2\big)$.

Proof. Take the scheme of Figure 1 with parameter ε in Lemma 2.5 and Theorem 3.2. Correctness and privacy follow from Lemmas 4.1 and 4.2. Its $\texttt{Setup}$ fixes $m = \lceil \sqrt{n} \rceil$ and $\beta = 3\lambda$, so $m^2 = O(n)$ and $\log m = \Theta(\log n)$. The word sizes $O(\log m)$ of Lemma 2.5 and $O(\log(m\beta))$ of Theorem 3.2 are $O(\log n + \log \lambda)$.

- *Preprocessing Runtime*: Step 2 of $\texttt{Prep}$ takes $O(m^{2+\varepsilon}) = O(n^{1+\varepsilon/2})$ time by Lemma 2.5. In Step 3, each h_i is a product of at most m integers of β bits, and a product tree computes it in $\widetilde{O}(m\beta)$ time, hence all h_i in $\widetilde{O}(m^2\beta)$ time. Theorem 3.2 with $\gamma = m\beta$, applied to each of the m rows, takes $m \cdot (1/\varepsilon)^{O(1/\varepsilon)} \cdot \widetilde{O}(\beta^{1+1/\varepsilon} m^{1+\varepsilon})$ time, which dominates both. As $m^{2+\varepsilon} = O(n^{1+\varepsilon/2})$ and $\beta = 3\lambda$, this is $(1/\varepsilon)^{O(1/\varepsilon)} \cdot \widetilde{O}(\lambda^{1+1/\varepsilon} \cdot n^{1+\varepsilon/2})$.
- *Server's Online Runtime*: In $\texttt{Answer}$, Step 3 takes $O(m^2/(\varepsilon \log m)^2)$ steps by Lemma 2.5, which is $O(n/(\varepsilon \log n)^2)$, and Step 4 takes $m \cdot \widetilde{O}(\beta m^\varepsilon/\varepsilon^2) = \widetilde{O}(\lambda \cdot n^{1/2+\varepsilon/2}/\varepsilon^2)$ time by Theorem 3.2 with $\gamma = m\beta$.
- *Client's Online Runtime*: $\texttt{Query}$ samples $(p, q) \leftarrow \texttt{Gen}(1^\lambda)$ in $T_{\texttt{Gen}}(\lambda)$ time and computes m greatest common divisors and m Legendre symbols of β-bit integers with respect to N and p, each in $\widetilde{O}(\beta)$ time by the fast Euclidean algorithm [vzGG13, Ch. 11] and the subquadratic algorithm of Brent and Zimmermann [BZ10]. $\texttt{Extract}$ reads the answer and computes one more Legendre symbol, for a total of $T_{\texttt{Gen}}(\lambda) + \widetilde{O}(m\beta) = T_{\texttt{Gen}}(\lambda) + \widetilde{O}(\sqrt{n} \cdot \lambda)$.
- *Online Communication*: The query consists of $\boldsymbol{v} \in \mathbb{Z}_2^m$ and $N < 2^{2\lambda}$, and the answer of $\boldsymbol{w} \in \mathbb{Z}_2^m$ and $\check{\boldsymbol{h}} \in \mathbb{Z}_N^m$, so $|\mathsf{q}| + |\mathsf{a}| = O(m\lambda) = O(\sqrt{n} \cdot \lambda)$.

□

References

BIM00. Amos Beimel, Yuval Ishai, and Tal Malkin. Reducing the servers computation in private information retrieval: PIR with preprocessing. In Mihir Bellare, editor, *CRYPTO 2000*, volume 1880 of *LNCS*, pages 55–73. Springer, Berlin, Heidelberg, August 2000. doi:10.1007/3-540-44598-6_4. 1

BIPW17. Elette Boyle, Yuval Ishai, Rafael Pass, and Mary Wootters. Can we access a database both locally and privately? In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part II*, volume 10678 of *LNCS*, pages 662–693. Springer, Cham, November 2017. doi:10.1007/978-3-319-70503-3_22. 2

BZ10. Richard P. Brent and Paul Zimmermann. An $O(M(n)\log n)$ algorithm for the Jacobi symbol. In *Algorithmic Number Theory*, Lecture Notes in Computer Science, pages 83–95. Springer, 2010. doi:10.1007/978-3-642-14518-6_10. 10

CGKS95. Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. Private information retrieval. In *36th FOCS*, pages 41–50. IEEE Computer Society Press, October 1995. doi:10.1109/SFCS.1995.492461. 1

CHR17. Ran Canetti, Justin Holmgren, and Silas Richelson. Towards doubly efficient private information retrieval. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part II*, volume 10678 of *LNCS*, pages 694–726. Springer, Cham, November 2017. doi:10.1007/978-3-319-70503-3_23. 2

CIJ+26. Caicai Chen, Yuval Ishai, Aayush Jain, Tamer Mour, Alon Rosen, and Chaoping Xing. Doubly-efficient secret-key PIR with low storage overhead. Cryptology ePrint Archive, Report 2026/1480, 2026. URL: https://eprint.iacr.org/2026/1480. 2

CIMR25. Caicai Chen, Yuval Ishai, Tamer Mour, and Alon Rosen. Secret-key PIR from random linear codes. Cryptology ePrint Archive, Report 2025/646, 2025. URL: https://eprint.iacr.org/2025/646. 2

CW25. Henry Corrigan-Gibbs and David J. Wu. The pseudorandomness of legendre symbols under the quadratic-residuosity assumption. In Benny Applebaum and Huijia (Rachel) Lin, editors, *TCC 2025, Part III*, volume 16270 of *LNCS*, pages 298–322. Springer, Cham, December 2025. doi:10.1007/978-3-032-12296-4_10. 3, 5

DMZ23. Xiuquan Ding, Giulio Malavolta, and Tianwei Zhang. Doubly efficient batched private information retrieval. Cryptology ePrint Archive, Report 2023/1552, 2023. URL: https://eprint.iacr.org/2023/1552. 2

GM82. Shafi Goldwasser and Silvio Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information. In *14th ACM STOC*, pages 365–377. ACM Press, May 1982. doi:10.1145/800070.802212. 2, 3

HHC+23. Alexandra Henzinger, Matthew M. Hong, Henry Corrigan-Gibbs, Sarah Meiklejohn, and Vinod Vaikuntanathan. One server for the price of two: Simple and fast single-server private information retrieval. In Joseph A. Calandrino and Carmela Troncoso, editors, *USENIX Security 2023*, pages 3889–3905. USENIX Association, August 2023. URL: https://www.usenix.org/conference/usenixsecurity23/presentation/henzinger. 3

KO97. Eyal Kushilevitz and Rafail Ostrovsky. Replication is NOT needed: SINGLE database, computationally-private information retrieval. In *38th FOCS*, pages 364–373. IEEE Computer Society Press, October 1997. doi:10.1109/SFCS.1997.646125. 1, 3

KU08. Kiran S. Kedlaya and Christopher Umans. Fast modular composition in any characteristic. In *49th FOCS*, pages 146–155. IEEE Computer Society Press, October 2008. doi:10.1109/FOCS.2008.13. 4, 6

Lee26. Keewoo Lee. Barely doubly-efficient SimplePIR. In Nadia Heninger and Mike Rosulek, editors, *Advances in Cryptology - CRYPTO 2026 - 46th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-20, 2026, Proceedings, Part X*, volume 16809 of *Lecture Notes in Computer Science*, pages 499–513. Springer, 2026. doi:10.1007/978-3-032-35428-0_16. 2, 3

LMW23. Wei-Kai Lin, Ethan Mook, and Daniel Wichs. Doubly efficient private information retrieval and fully homomorphic RAM computation from ring LWE. In Barna Saha and Rocco A. Servedio, editors, *55th ACM STOC*, pages 595–608. ACM Press, June 2023. doi:10.1145/3564246.3585175. 2, 4

LMW25. Wei-Kai Lin, Ethan Mook, and Daniel Wichs. Black box crypto is useless for doubly efficient PIR. In Serge Fehr and Pierre-Alain Fouque, editors, *EUROCRYPT 2025, Part VI*, volume 15606 of *LNCS*, pages 65–93. Springer, Cham, May 2025. doi:10.1007/978-3-031-91095-1_3. 2

LPR10. Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 1–23. Springer, Berlin, Heidelberg, May / June 2010. doi:10.1007/978-3-642-13190-5_1. 2

OPPW24. Hiroki Okada, Rachel Player, Simon Pohmann, and Christian Weinert. Towards practical doubly-efficient private information retrieval. In Jeremy Clark and Elaine Shi, editors, *FC 2024, Part II*, volume 14745 of *LNCS*, pages 264–282. Springer, Cham, March 2024. doi:10.1007/978-3-031-78679-2_14. 2

OPPW25. Hiroki Okada, Rachel Player, Simon Pohmann, and Christian Weinert. On algebraic homomorphic encryption and its applications to doubly-efficient PIR. In Serge Fehr and Pierre-Alain Fouque, editors, *EUROCRYPT 2025, Part VI*, volume 15606 of *LNCS*, pages 34–64. Springer, Cham, May 2025. doi:10.1007/978-3-031-91095-1_2. 2

Reg05. Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th ACM STOC*, pages 84–93. ACM Press, May 2005. doi:10.1145/1060590.1060603. 2

vzGG13. Joachim von zur Gathen and Jürgen Gerhard. *Modern Computer Algebra*. Cambridge University Press, 3 edition, 2013. doi:10.1017/CBO9781139856065. 7, 10

Wil07. Ryan Williams. Matrix-vector multiplication in sub-quadratic time: (some preprocessing required). In Nikhil Bansal, Kirk Pruhs, and Clifford Stein, editors, *18th SODA*, pages 995–1001. ACM-SIAM, January 2007. 3, 5

XZO+26. Pan Xiao, Heng Zhang, Rending Ouyang, Cong Zhang, Jian Liu, Kui Ren, and Chun Chen. Towards making doubly-efficient PIR practical. Cryptology ePrint Archive, Report 2026/243, 2026. URL: https://eprint.iacr.org/2026/243. 2