Improved Performance for Private Information Retrieval

Boyan Litchev — MIT PRIMES

Simon Langowski (Mentor) — MIT

June 2024

By allowing users to retrieve items from a database without revealing which item was retrieved, Private Information Retrieval (PIR) has enabled recent advances in anonymous communication, private streaming, and more. However, PIR is very computationally expensive, and is fundamentally limited to having a computational cost that scales linearly with the size of the database, limiting the scale of protocols that use it to millions of users. By adjusting the procedure for gadget inversions, a key step in the homomorphic multiplications used in PIR, we achieve a 30% speedup over existing state-of-the-art PIR protocols and similarly reduce network costs.

1 Introduction

By allowing users to retrieve an item from a database without revealing which item was requested to the server which contains the database, Private Information Retrieval (PIR) has been crucial to recent advances in anonymous communication [2, 5], anonymous streaming [15], and other privacy-preserving services.

To do this, PIR has users send a query to the server, which utilizes the query to retrieve the desired element and send it to the client. To ensure that the query does not reveal the requested element to the server, while still allowing the server to send just the requested element to the client, PIR utilizes homomorphic encryption (HE) [7, 9, 13, 21, 14] to perform operations on an encryption of the database index the client wishes to retrieve.

Due to the large ciphertext sizes and computation times of current HE schemes, Private Information Retrieval has large network and computational costs, preventing its widespread adoption.

Information-theoretic PIR [11, 12] decreases these costs by assuming that multiple non-colluding servers have a copy of the database, and having the client send a query to each of them. However, assuming that multiple servers will not collude is often impractical.

Computational PIR (CPIR) makes no non-collusion assumptions, so we focus on it in this work. Some recent advances in CPIR preprocess the database [6, 8], or have the client pre-compute database-dependent hints [20, 19] in order to reduce the server-side costs of PIR. These types of schemes rely on expensive, database-dependant computations being amortized over many queries. In applications such as anonymous messaging, the database of messages can change frequently, meaning that the pre-computations cannot be amortized.

Single-round CPIR protocols that do not perform any database or query-dependent computations [4, 17, 1, 18] often have their computation times dominated by many large homomorphic multiplications. Each homomorphic ciphertext has an error associated with it, and in order for the user of a PIR scheme to successfully retrieve their desired database element, this error cannot exceed a certain threshold. The threshold can be increased by having larger ciphertexts, on which operations are performed more slowly. Since the error grows multiplicatively after homomorphic multiplications, large ciphertexts are used, resulting in slower multiplications.

To reduce this error, Spiral[17] utilizes gadget inversions to reduce the error after each multiplication. In 3-PIR, we modify this process by increasing the base of the gadget inversion to 3, which reduces both the cost of a gadget multiplication, and the error after a gadget multiplication, resulting in smaller ciphertext sized being used and even more computational savings.

So, in summary 3-PIRincreases the base of gadget inversions to 3 in order to

2 Background

We now formally introduce the homomorphic encryption used to build Private Information Retrieval (PIR), and some details about PIR. Our work primarily focuses on modifying Spiral [17], so this background is specific to the homomorphic operations and PIR techniques used in it.

2.1 Homomorphic Encryption

Homomorphic encryption [7, 9, 13, 21, 14, 10] allows for operations (in this case, addition and multiplication) to be performed on encrypted elements. The schemes we consider have their security based on ring learning with errors [16], and are based on multiplications of elements of the ring $\mathbb{Z}_q[X]/(X^n + 1)$.

Elements of the ring $\mathbb{Z}_q[X]/(X^n+1)$ will be denoted with lowercase letters, and referred to as "polynomials". Matrices of polynomials will be denoted with uppercase letters. The vertical concatenation of A and B will be denoted as $\begin{bmatrix} A \\ B \end{bmatrix}$, and the horizontal concatenation will be denoted as $\begin{bmatrix} A & B \end{bmatrix}$. Finally, the letters $q, n, l = \lfloor \log_2(q) \rfloor$, and t, will denote the parameters of homomorphic encryption; q will be referred to as the modulus, n as the polynomial length, and t as the plaintext modulus.

Ciphertext Representation. Every homomorphic ciphertext has some error e that is randomly drawn from a distribution, is encrypted with some secret key s chosen by the user, and stores a message m. A generic ciphertext is represented as the matrix.

$$C = \begin{bmatrix} a \\ as + e + tm \end{bmatrix} = \begin{bmatrix} c_0 \\ c_1 \end{bmatrix}$$

Under the ring learning with errors assumption [16], an adversary (in the case of PIR, the server), cannot learn what the message m is, provided that q, n, and t satisfy certain conditions, which can be checked with a RLWE security estimator [3]. Note that this ciphertext depends on a parameter t, the plaintext modulus, and that t is an integer, while a, s, e, and m are polynomials. To decrypt a ciphertext, the user can compute

$$\text{Dec}(X) = \text{Dec} \begin{bmatrix} c_0 \\ c_1 \end{bmatrix} = c_1 - sc_0 = (as + e + tm) - s(a) = e + tm$$

Then, to retrieve the message the user computes $\frac{\text{Dec}(X)}{t}$, and rounds the result the nearest integer. We will make two important notes: firstly, the successful decryption of the ciphertext only occurs if $e < t$, so it must be ensured that the error does not grow too large during computations; and secondly, the decryption function is linear, as

$$\text{Dec}(aX + bY) = a\text{Dec}(X) + b\text{Dec}(Y)$$

Adding the message of two ciphertexts X and Y can be performed by the server (which does not have access to the secret key) by computing $X + Y$ (matrix addition), since the decryption function is linear. This also increases the error additively.

Multiplication. In multiplicatively homomorphic schemes such as BGV [7] and BFV [13], the server can multiply two ciphertexts, at the cost of a multiplicative increase in error. After successive multiplications, this causes the error to rapidly increase above t, so to compensate some schemes set a large plaintext modulus

In the specific case of Private Information Retrieval (PIR), all multiplications consist of multiplying a ciphertext by an encryption of 0 or an encryption of 1. Because of this, a technique called gadget inversions is employed.

Gadget Inversions are used to reduce the amount of error after a homomorphic multiplication, so that error grows additive after multiplications. In a gadget multiplication, the object which encrypts 0 or 1 no longer has the same dimensions as a ciphertext; instead we refer to it as a query ciphertext, or query. The product of a ciphertext and gadget ciphertext yields a ciphertext. We now outline a gadget multiplication below.

For simplicity, we denote $\lfloor \log_2 q \rfloor$ as l, where q is the ciphertext modulus. In addition, let

$$G = \begin{bmatrix} 1 & 2 & \cdots & 2^l \end{bmatrix}$$

If an arbitrary polynomial a's binary representation is $\overline{a_l a_{l-1} \dots a_1 a_0}$, such that $a = 2^l a_l + 2^{l-1} a_{l-1} + \dots + 2^1 a_1 + 2^0 a_0$, and each polynomial a_i has coefficients that are 0 or 1, then

$$G^{-1}(a) = \begin{bmatrix} a_l \\ a_{l-1} \\ \vdots \\ a_1 \\ a_0 \end{bmatrix}$$

$$G^{-1}\left(\begin{bmatrix} a \\ b \end{bmatrix}\right) = \begin{bmatrix} G^{-1}(a) \\ G^{-1}(b) \end{bmatrix}$$

Note that by definition, $G \cdot G^{-1}(a) = a$.

Query ciphertexts can either encrypt 0 or 1, as follows:

$$\text{Enc}(0) = \begin{bmatrix} b_0 & b_1 & \dots & b_{2l-2} & b_{2l-1} \\ b_0 s + e_0 & b_1 s + e_1 & \dots & b_{2l-2} s + e_{2l-2} & b_{2l-1} s + e_{2l-1} \end{bmatrix}$$

$$\text{Enc}(1) = \text{Enc}(0) + \begin{bmatrix} G & 0 \\ 0 & G \end{bmatrix}$$

To multiply a ciphertext $C = \begin{bmatrix} a \\ as + e + tm \end{bmatrix}$ by a query Q, we compute $Q \cdot G^{-1}(C)$.

If $Q = \text{Enc}(0)$, then the product evaluates to

$$\begin{aligned} P &= \begin{bmatrix} b_0 & b_1 & \dots & b_{2l-2} & b_{2l-1} \\ b_0 s + e_0 & b_1 s + e_1 & \dots & b_{2l-2} s + e_{2l-2} & b_{2l-1} s + e_{2l-1} \end{bmatrix} \cdot G^{-1}(C) \\ &= \begin{bmatrix} b_0 & b_1 & \dots & b_{2l-2} & b_{2l-1} \\ b_0 s + e_0 & b_1 s + e_1 & \dots & b_{2l-2} s + e_{2l-2} & b_{2l-1} s + e_{2l-1} \end{bmatrix} \cdot \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_{2l-2} \\ c_{2l-1} \end{bmatrix} \\ &= \begin{bmatrix} \sum_{i=0}^{2l-1} b_i c_i \\ \sum_{i=0}^{2l-1} (b_i s + e_i) c_i \end{bmatrix} \end{aligned}$$

So,

$$\text{Dec}(P) = \sum_{i=0}^{2l-1} (b_i s + e_i) c_i - s \cdot \sum_{i=0}^{2l-1} b_i c_i = \sum_{i=0}^{2l-1} e_i c_i$$

, which decrypts to 0. Since all the coefficients of c_i are 0 or 1, $c_i e_i$ has coefficients that are at most n times the e_i, for all i. So, the error has increased additively by a number that is at most $2ln$ times an error from the original error distribution.

If $Q = \text{Enc}(1)$, then the product evaluates to

$$\begin{aligned} P' &= \left(\text{Enc}(0) + \begin{bmatrix} G & 0 \\ 0 & G \end{bmatrix}\right) \cdot G^{-1}(C) \\ &= \text{Enc}(0) \cdot G^{-1}(C) + \begin{bmatrix} G & 0 \\ 0 & G \end{bmatrix} G^{-1}(C) \\ &= \begin{bmatrix} \sum_{i=0}^{2l-1} b_i c_i \\ \sum_{i=0}^{2l-1} (b_i s + e_i) c_i \end{bmatrix} + \begin{bmatrix} a \\ as + e + tm \end{bmatrix} = \begin{bmatrix} a + \left(\sum_{i=0}^{2l-1} b_i c_i\right) \\ as + e + tm + \left(\sum_{i=0}^{2l-1} (b_i s + e_i) c_i\right) \end{bmatrix} \end{aligned}$$

So,

$$\text{Dec}(P') = e + tm + \text{Dec}(P) = \sum_{i=0}^{2l-1} e_i c_i$$

, which decrypts to the original message m, and the error in the ciphertext has again increased by at most $2ln$ times a value from the original error distribution.

In a gadget multiplication, more polynomials have to be multiplied than in the ciphertext multiplications in other HE schemes. However, due to the decreased error from gadget multiplications, when using them the parameters q, n, and t can be decreased, resulting in faster computation.

2.2 Private Information Retrieval

Private information retrieval (PIR) aims to retrieve a database item without revealing which item was retrieved. Trivially, this could be done by sending the entire database to a user, thus transmitting the desired item. However, this would result in a prohibitively large network overhead, since the response size would be the size of the database.

So, PIR aims to decrease network costs by compressing the d-element database into a ciphertext that contains the information of the desired database index. Traditionally, this is done using three operations – the client can generate a query or decode a response, and the server must have a procedure to "answer" a query by generating a response that can be decrypted by the client. Formally, there are three procedures such that:

$$\begin{aligned} \text{Query}(\text{idx}) &= \text{query} \\ \text{Answer}(\text{query}, \text{db}) &= \text{response} \\ \text{Decode}(\text{response}) &= \text{database}[\text{index}] \end{aligned}$$

In practice, the answer procedure is the most expensive: in order to hide which database element was retrieved, all database elements must be involved in the computation process–if any element was not involved, it would leak information about the retrieved index. Because of this, we will focus on how the server compresses the database down to a smaller ciphertext.

Linear Private Information Retrieval accomplishes the goal of compression using a one-hot query $Q = (c_0, c_1, \dots, c_i, \dots, c_{d-1})$, where c_i encodes a 1 for the desired database index and 0 for all other indices. The database $D = (p_0, p_1, \dots, p_{d-1})$ is then multiplied by the query to produce the result $R = \sum_{i=0}^{d-1} c_i p_i$. For all non-desired elements, $c_i p_i = 0$ since $c_i = 0$, and for the desired element $c_i p_i = p_i$ as $c_i = 1$, so this sum generates a ciphertext encoding just the desired element.

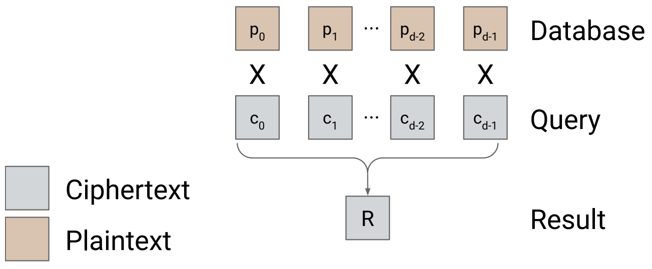

Figure 1: A linear PIR scheme. Note that homomorphic encryption is used so that operations can be performed using the query values.

Two-dimensional databases can reduce the query size while maintaining a relatively fast answer procedure, as shown in Figure 2. Formally, if we have a database $D = (p_{0,0}, p_{0,1}, \dots, p_{0,m-1}, p_{1,0}, p_{1,1}, \dots, p_{1,m-1}, \dots p_{n-1,m-1}$ with $mn = d$, then we can have query $Q = (c_{0,0}, c_{0,1}, \dots c_{0,m-1}, c_{1,0}, c_{1,1}, \dots c_{1,n-1})$, where both $(c_{0,0}, \dots c_{0,m-1})$ and $(c_{1,0}, \dots c_{1,n-1})$ are one-hot vectors. Then, the result $R = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} c_{0,i} c_{1,j} p_{i,j}$ is computed, and it encodes the desired element, since all other elements are multiplied by zero.

Note that in practice this is computed as $R = \sum_{i=0}^{n-1} c_{0,i} \left(\sum_{j=0}^{m-1} c_{1,j} p_{i,j} \right)$, meaning $d + n$ multiplications (and not $2d$) are now required to compute the result. In addition, the two-dimensional structure means that in an m by n database, only $m + n$ expanded query ciphertexts are needed instead of mn.

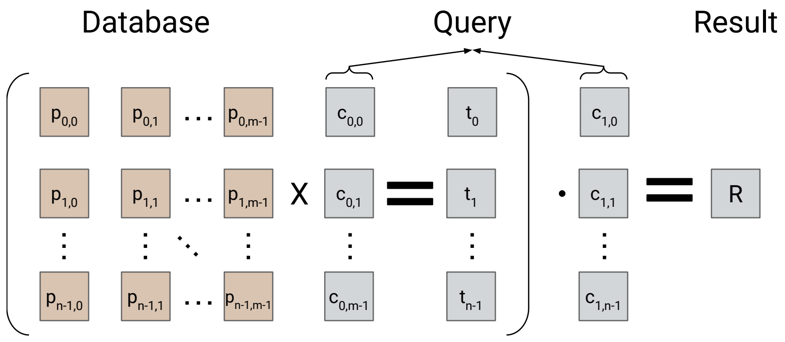

Figure 2: Reformatting the database into a rectangular matrix results in a smaller query at the cost of slightly more multiplications.

Higher Dimensions of the database eventually result in a procedure we call "**folding**", where each dimension has size two, as shown in Figure 3. Since the query for each dimension is a one-hot vector, one of the two ciphertexts will always be an encoding of 0, and the other will always encode 1. This means one of the ciphertexts is always 1 minus the other, so in practice only one ciphertext is sent. Therefore, the query length for a folding PIR protocol is only $\log_2(d)$, and from a computational perspective $2d$ multiplications are required.

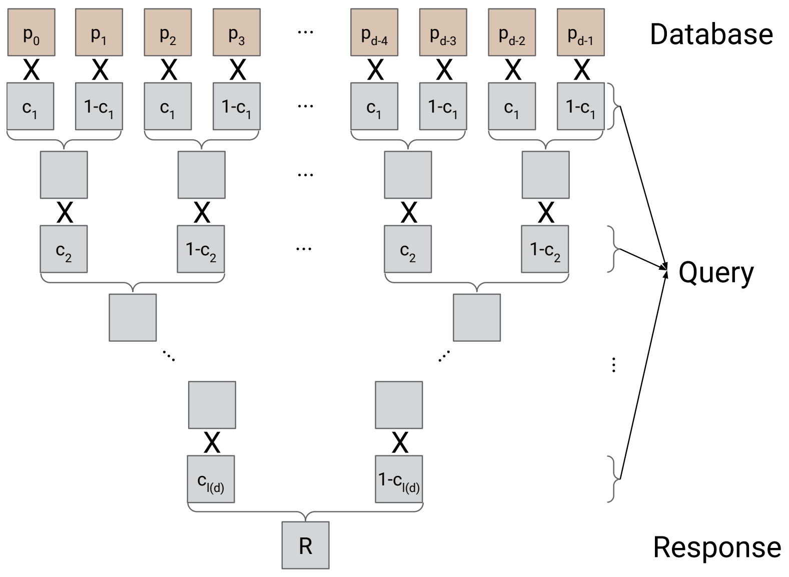

Figure 3: "Folding": each ciphertext folds the database in half

Such a scheme is partially used by Spiral [17], which has one large dimension and then many small dimensions of size two.

3 3-PIR

We now introduce 3-PIR, which consists of the same protocol as Spiral [17], with a gadget procedure modified to reduce the amount of error and computation after each multiplication.

3.1 Base-3 Gadget Decompositions

Traditionally, a gadget multiplication has

$$G = \begin{bmatrix} 1 & 2 & \cdots & 2^l \end{bmatrix}$$

, and G^{-1} performs a binary decomposition by decomposing a polynomial into polynomials with coefficients that are 0 or 1.

Instead, we let

$$G_3 = \begin{bmatrix} 1 & 3 & \cdots & 3^{l'} \end{bmatrix}$$

, and have G_3^{-1} perform a balanced ternary decomposition. Specifically, if an arbitrary polynomial a's balanced ternary representation is $\overline{a_{l'}a_{l'-1}\ldots a_1 a_0}$, such that $a = 3^{l'}a_{l'} + 3^{l-1}a_{l'-1} + \ldots + 3^1 a_1 + 2^0 a_0$, where all a_i are polynomials with coefficients that are -1, 0, or 1, then

$$G_3^{-1}(a) = \begin{bmatrix} a_{l'} \\ a_{l'-1} \\ \vdots \\ a_1 \\ a_0 \end{bmatrix}$$

$$G_3^{-1}\left(\begin{bmatrix} a \\ b \end{bmatrix}\right) = \begin{bmatrix} G_3^{-1}(a) \\ G_3^{-1}(b) \end{bmatrix}$$

As before, $G_3 \cdot G_3^{-1}(a) = a$. The advantage of this approach is that now l' only needs to be $\lfloor \log_3(2q) \rfloor$ so that we can represent every possible polynomial a, which is smaller than $l = \lfloor \log_2(q) \rfloor$.

We define the two query ciphertexts Enc(0) and Enc(1) as before, except that they now have l' columns instead of l columns. These changes have two primary effects:

Firstly, to compute the product of a query ciphertext and a ciphertext, we now only have to compute $2l'^2$ polynomial products instead of $2l^2$, since the matrices of polynomials that we multiply are smaller.

Secondly, after each multiplication, the error now increases by $2l'$ times an error from the original error distribution, instead of $2l$ times.

4 Evaluation

We implement our protocol as clone of Spiral [17], with modifications to the gadget process.

4.1 Per-multiplication costs

Firstly, we examine the costs of each query-ciphertext by ciphertext multiplication with our new gadget procedure.

With the modifications of 3-PIR, computing G^{-1} takes more time than before, since a trinary decomposition of the ciphertext polynomials cannot be performed with bit shifts, as opposed to the original binary decomposition. However, as a result of the decreased number of polynomial multiplications that need to be performed, the total time to perform a query ciphertext by ciphertext multiplication decreases significantly.

In addition to the speedup of base-3 gadgets multiplications, 3-PIRcombines the gadget inversion and polynomial multiplication steps of the query ciphertext by ciphertext multiplication into a singular step in order to reduce the number of memory accesses.

	Gadget Inversion (μs)	Other Costs (μs)	Total Time (μs)
Base-2 Gadget (Spiral)	560	4580	5140
Base-3 Gadget	1280	2590	3870
Optimized Base-3 Gadget	*	*	2850
Speedup Over Spiral	0.5x*	1.77x*	1.8x

Figure 4: The cost of performing a query ciphertext by ciphertext multiplication for Spiral, and our scheme. All trials were run on my laptop (6-core AMD Ryzen 5 5500U, 20GB RAM, 2.10 GHz) using a single core. *Since the optimized base-3 gadget cannot be broken down into its component steps, speedups are reported for the non-optimized version.

This results in the "Optimized Base-3 Gadget" which is nearly 1.8x faster than the multiplications of Spiral.

5 Conclusion

By using a modified gadget multiplication, 3-PIRprovides a concrete speedup over other leading Private Information Retrieval schemes, without leading to larger network costs.

In the future, we would like to further optimize this protocol by

- Considering alternate gadget decomposition methods–different possibilities for the matrix G and a corresponding function G^{-1} such that $GG^{-1}(a) = a$.
- Modifying parameter sets to reduce ciphertext sizes
- Examining the effects of our protocol on larger databases
- Examining the effects of the revised multiplication on the time for an entire PIR answer.

6 Acknowledgments

I would like to thank Simon Langowski for his guidance and support of this project, and MIT PRIMES for making this project possible.

References

[1] Carlos Aguilar Melchor, Joris Barrier, Laurent Fousse, and Marc-Olivier Killijian. XPIR : Private Information Retrieval for Everyone. *Proceedings on Privacy Enhancing Technologies*, avril 2016:155–174, April 2016.

[2] Ishtiyaque Ahmad, Yuntian Yang, Divyakant Agrawal, Amr El Abbadi, and Trinabh Gupta. Addra: Metadata-private voice communication over fully untrusted

infrastructure. In *15th USENIX Symposium on Operating Systems Design and Implementation (OSDI 21)*, pages 313–329. USENIX Association, July 2021.

[3] Martin R. Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of learning with errors. Cryptology ePrint Archive, Paper 2015/046, 2015. https://eprint.iacr.org/2015/046.

[4] Sebastian Angel, Hao Chen, Kim Laine, and Srinath Setty. Pir with compressed queries and amortized query processing. Cryptology ePrint Archive, Paper 2017/1142, 2017. https://eprint.iacr.org/2017/1142.

[5] Sebastian Angel and Srinath Setty. Unobservable communication over fully untrusted infrastructure. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 551–569, Savannah, GA, November 2016. USENIX Association.

[6] Amos Beimel, Yuval Ishai, and Tal Malkin. 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 20*, pages 55–73. Springer, 2000.

[7] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. Fully homomorphic encryption without bootstrapping. Cryptology ePrint Archive, Paper 2011/277, 2011. https://eprint.iacr.org/2011/277.

[8] Ran Canetti, Justin Holmgren, and Silas Richelson. Towards doubly efficient private information retrieval. In *Theory of Cryptography: 15th International Conference, TCC 2017, Baltimore, MD, USA, November 12-15, 2017, Proceedings, Part II 15*, pages 694–726. Springer, 2017.

[9] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic encryption for arithmetic of approximate numbers. Cryptology ePrint Archive, Paper 2016/421, 2016. https://eprint.iacr.org/2016/421.

[10] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. Tfhe: fast fully homomorphic encryption over the torus. *Journal of Cryptology*, 33(1):34–91, 2020.

[11] Ben-Zion Chor, Oded Goldreich, and Eyal Kushilevitz. Private information retrieval, December 29 1998. US Patent 5,855,018.

[12] Daniel Demmler, Amir Herzberg, and Thomas Schneider. Raid-pir: practical multi-server pir. In *Proceedings of the 6th edition of the ACM Workshop on Cloud Computing Security*, pages 45–56, 2014.

[13] Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Paper 2012/144, 2012. https://eprint.iacr.org/2012/144.

[14] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *Advances in Cryptology–CRYPTO 2013: 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, pages 75–92. Springer, 2013.

[15] Trinabh Gupta, Natacha Crooks, Whitney Mulhern, Srinath Setty, Lorenzo Alvisi, and Michael Walfish. Scalable and private media consumption with popcorn. In *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16)*, pages 91–107, Santa Clara, CA, March 2016. USENIX Association.

[16] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In *Advances in Cryptology–EUROCRYPT 2010: 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30–June 3, 2010. Proceedings 29*, pages 1–23. Springer, 2010.

[17] Samir Jordan Menon and David J. Wu. Spiral: Fast, high-rate single-server pir via fhe composition. Cryptology ePrint Archive, Paper 2022/368, 2022. `https://eprint.iacr.org/2022/368`.

[18] Muhammad Haris Mughees, Hao Chen, and Ling Ren. Onionpir: Response efficient single-server pir. Cryptology ePrint Archive, Paper 2021/1081, 2021. `https://eprint.iacr.org/2021/1081`.

[19] Hoang-Dung Nguyen, Jorge Guajardo, and Thang Hoang. Client-efficient online-offline private information retrieval. Cryptology ePrint Archive, Paper 2024/719, 2024. `https://eprint.iacr.org/2024/719`.

[20] Sarvar Patel, Giuseppe Persiano, and Kevin Yeo. Private stateful information retrieval. In *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*, pages 1002–1019, 2018.

[21] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM)*, 56(6):1–40, 2009.