Private Information Retrieval in Large Scale Public Data Repositories

Ishtiyaque Ahmad
UC Santa Barbara
ishtiyaque@ucsb.edu

Divyakant Agrawal
UC Santa Barbara
agrawal@cs.ucsb.edu

Amr El Abbadi
UC Santa Barbara
amr@cs.ucsb.edu

Trinabh Gupta
UC Santa Barbara
trinabh@ucsb.edu

ABSTRACT

The tutorial focuses on Private Information Retrieval (PIR), which allows clients to privately query public or server-owned databases without disclosing their queries. The tutorial covers the basic concepts of PIR such as its types, construction, and critical building blocks, including homomorphic encryption. It also discusses the performance of PIR, existing optimizations for scalability, real-life applications of PIR, and ways to extend its functionalities.

PVLDB Reference Format:
Ishtiyaque Ahmad, Divyakant Agrawal, Amr El Abbadi, and Trinabh Gupta. Private Information Retrieval in Large Scale Public Data Repositories. PVLDB, 16(12): 3868 - 3871, 2023.
doi:10.14778/3611540.3611572

1 INTRODUCTION

As more data moves to the cloud, privacy protection is becoming increasingly crucial in the field of databases. While much of the research in this area has focused on encryption-based solutions for *private* databases, relatively little attention has been paid to provide privacy for querying *public* or server-controlled databases where a user cannot encrypt the data. This gap is where Private Information Retrieval (PIR) [14, 15, 22] can play a crucial role. PIR allows users to privately query a database without revealing any information about the query, making it ideal for publicly available or server-owned databases. Although PIR has limitations in terms of performance and scalability, extensive research has been done since its inception in 1995 [14] to address these issues. PIR has now been applied in many real-life applications such as anonymous communication [4, 6], content sharing [19], and keyword search [3, 31]. With the rise of big data and the increasing importance of privacy protection, there is a growing need for research in this area.

Tutorial overview: This tutorial is 1.5 hours long and presented in a lecture style. It covers the basics of Private Information Retrieval (PIR), including its fundamentals, types, algorithm construction, and performance optimization. It also explores various ways to extend the functionalities of PIR and highlights potential areas for future research. The target audience is researchers from academia and industry. The tutorial is designed to be beginner-friendly, starts with basic concepts, and does not require any specialized background knowledge. The tutorial is organized as follows:

(1) **Introduction to Private Information retrieval (15 mins)**.
(2) **A primer on homomorphic encryption (15 mins)**.
(3) **Computational PIR: construction and analysis (40 mins)**.
(4) **Extension of PIR: key-value retrieval (15 mins)**.
(5) **Conclusion and future research (5 mins)**

Proceedings of the VLDB Endowment, Vol. 16, No. 12 ISSN 2150-8097.
doi:10.14778/3611540.3611572

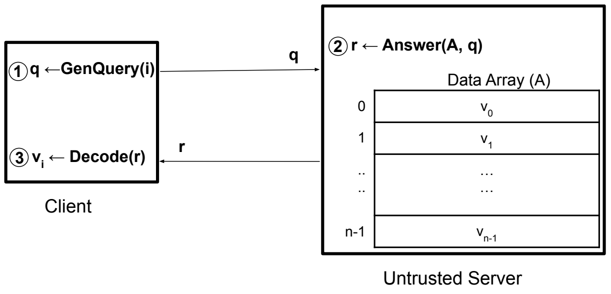

Figure 1: **A schematic diagram illustrating the three steps of the Private Information Retrieval (PIR) protocol.**

2 INTRODUCTION TO PRIVATE INFORMATION RETRIEVAL (PIR)

Private Information Retrieval (PIR) [14, 15, 22] enables a client to retrieve an element at a specific array index from an untrusted server without revealing which element was retrieved. A PIR protocol takes place between a PIR server and a PIR client as illustrated in Figure 1. The server owns an array $\mathcal{A}$ of n elements, and the client wants to retrieve the element at a specific index i without revealing i or $\mathcal{A}[i]$ to anyone, including the server. The PIR protocol comprises three steps. In step 1, the client generates a query q that conceals the desired index i and sends q to the server. In step 2, the server computes an answer r and sends it back to the client. Finally, in step 3, the client decodes r to get the desired element $\mathcal{A}[i]$. A PIR protocol must provide the following two guarantees:

(1) **Correctness:** If a client requests the element at index i in the array $\mathcal{A}$, then the protocol must provide it with $\mathcal{A}[i]$.
(2) **Privacy:** The PIR server must not learn any information about i or $\mathcal{A}[i]$. Specifically, if the PIR client makes two separate PIR queries with indices i and j, then the server must not be able to distinguish between i and j with a non-negligible probability.

A strawman approach to PIR is for the client to download the entire array $\mathcal{A}$ and then access the element at index i locally. This approach is often referred to as *trivial PIR* in literature. However, this scheme is not scalable as the response size grows with the size of $\mathcal{A}$. Therefore, we are particularly interested in solutions where the response size is a function of the size of a single array element rather than the entire array.

There are two main categories of private information retrieval protocols: information theoretic PIR (IT-PIR) [14, 15] and computational PIR (CPIR) [22]. IT-PIR replicates the array $\mathcal{A}$ across multiple servers and each server generates a share of the PIR response which the client can combine to obtain the desired element. This multi-server PIR protocols make the assumption that the servers do not collude among each other and thus provide information theoretic security. However, the requirement of non-colluding servers in IT-PIR can be difficult to achieve in practice. CPIR, on the other hand, stores $\mathcal{A}$ on a single server and provides cryptographic security to its clients. CPIR protocols use *homomorphic encryption* [18, 26] to conceal the client's query and response from the server. This tutorial focuses on single server CPIR protocols, which are more practical to deploy because they do not require non-colluding servers. However, a significant challenge of CPIR protocols is that they incur higher computational overheads.Much of the recent innovations in this field have been to develop techniques that reduce the costs of homomorphic encryption, and hence making PIR practical in many realistic large scale data management applications.

3 A PRIMER ON HOMOMORPHIC ENCRYPTION

Homomorphic encryption allows computation over encrypted data. As a result, a client can outsource computation to an untrusted cloud without revealing its private data. The client starts by generating a *secret key, sk*, and a *public key, pk*, following the parameters of a particular homomorphic encryption scheme. In this tutorial, we will discuss the following operations.

- **Encrypt**(v, pk) encrypts plaintext message v using *public key pk* to produce ciphertext c. Generally, the size of the resulting ciphertext, c, is greater than v. The ratio $\frac{size(c)}{size(v)}$ is known as the *expansion factor* (F) of the homomorphic encryption scheme.
- **Decrypt**(c, sk) decrypts the ciphertext c using the *secret key* to recover the original plaintext message v.
- **Add**(c_1, c_2) takes two ciphertexts c_1 and c_2 as input which are encryptions of plaintexts v_1 and v_2 respectively, and outputs an encryption of $(v_1 + v_2)$.
- **Multiply**(c_1, c_2) takes two ciphertexts c_1 and c_2 as input which are encryptions of plaintexts v_1 and v_2 respectively, and outputs an encryption of $(v_1 * v_2)$.
- **MultiplyPlain**(v_1, c_2) takes a plaintext v_1 and a ciphertext c_2 which is the encryption of a plaintexts v_2, and outputs an encryption of $(v_1 * v_2)$.

Homomorphic encryption schemes that support a subset of the above homomorphic operations are called Partially Homomorphic Encryption. They can be either *additive*, supporting Add and MultiplyPlain (e.g., Paillier [26]), or *multiplicative*, supporting Multiply over encrypted data (e.g., El Gamal [16]).

Fully Homomorphic Encryption (FHE) is the most powerful type of homomorphic encryption scheme that supports all the operations discussed above. In theory, Fully Homomorphic Encryption (FHE) can perform any linear operation over encrypted data. However, the computations in FHE can be significantly more expensive than the corresponding operations over plaintext data. The first FHE scheme was developed by Gentry [18] in 2009 using lattice-based cryptography. Subsequently, several other schemes have been introduced, such as BFV [8, 17], BGV [9], CKKS [11], TFHE [12], and more, improving the performance by several orders of magnitude. These schemes leverage lattice-based operations, which are faster than traditional number-theoretic encryptions like Paillier and El Gamal. Therefore, in certain scenarios, it may be preferable to use a subset of FHE operations instead of a specifically additive or multiplicative homomorphic encryption.

4 COMPUTATIONAL PIR: CONSTRUCTION AND ANALYSIS

This section describes how we can build a Computational Private Information retrieval (CPIR) [22] solution using additive homomorphism of an encryption scheme.

A CPIR query comprises an array of n additive homomorphic ciphertexts, where n is the total number of elements in $\mathcal{A}$ at the server. The ciphertext at the desired index i is an encryption of 1, and the rest are encryptions of 0. Due to the non-deterministic property of Encrypt (§3), all the encryptions of 0 will be distinct. To generate a response, the server first uses the MultiplyPlain (§3) function to multiply each ciphertext in the query with the corresponding element in $\mathcal{A}$. The response of a CPIR protocol is the sum of the outputs of these MultiplyPlain operations, obtained using the Add procedure. This results in an encryption of the element at index i. The client can then Decrypt the response to obtain the desired element $\mathcal{A}[i]$. Note that, to answer a single PIR query, the server needs to process all the elements in the database array $\mathcal{A}$. This is a fundamental lower bound of computation for PIR [7] because if the server can exclude processing a subset of the array, it will know that the client's query does not involve that part of the array, thus violating the privacy guarantee of PIR (§2). This computation bound is a key challenge to the performance and scalability of CPIR-based solutions.

There are three key performance metrics to consider when evaluating a PIR scheme: i) the size of the query, ii) the computation time for response generation, and iii) the size of the response. These metrics are often in conflict with each other, meaning that improving one may lead to a reduced performance in another. Therefore, optimizing CPIR performance requires research on multiple fronts. A popular technique, proposed by Stern [30], to address the issue of query-size is called *recursion*. This approach involves arranging the data array into a d-dimensional hypercube and performing PIR along each dimension sequentially. This reduces the query size to $d \cdot \sqrt[d]{n}$ ciphertexts instead of n. However, after each dimension, the output is expanded by the *expansion factor* F (§3) of the encryption scheme. Consequently, the final PIR response-size gets expanded by a factor of F^d, making it impractical to use a large value of d.

This tutorial discusses three popular open-source CPIR schemes. XPIR [1] was the first to use lattice-based cryptography to build a CPIR solution, resulting in significant improvements over prior works. SealPIR [5] developed a new query compression technique that reduces query size significantly, but it comes with an additional server-side computational cost to expand queries. FastPIR [4] reduces query size compared to XPIR by using the vectorized form of BFV [8, 17] and taking advantage of the Single Instruction Multiple Data (SIMD) technique. It also reduces computation time by applying cryptographic optimizations, but at the expense of a compromise in query size compared to SealPIR.

4.1 Retrieving Multiple Elements

Suppose a client wants to retrieve $k \geq 2$ elements from an untrusted server privately. One natural approach would be for the client to send k PIR queries. However, this approach is inefficient, as it increases the query size, response generation time, and response size by a factor of k. Instead, batch codes [21] can be used to reduce the computation cost at the server. A batch code is an encoding mechanism that takes n elements from a database as input and generates a set of m codewords distributed among b buckets, where $b > k$ and $n < m < kn$. The key feature of batch codes is that any k elements from the original database can be retrieved privately by fetching one codeword from each of the b buckets using PIR. This incurs a computation cost of $O(m)$, which is less expensive than the $O(kn)$ cost for k separate PIR queries over n elements. However, the response size is increased to b ciphertexts instead of k ciphertexts. One major drawback of batch codes is that for large n and k, the response size b can become prohibitively large.

Angel et. al. [5] addressed this issue by proposing a new encoding method named *probabilistic batch codes* (PBC). The PBC technique keeps the number of bins b as a small multiple of k (e.g., $b = 1.5k$). As a result, the response size is inflated by a small amount. However, due to the probabilistic nature of their algorithm, there is a small probability ($\approx 2^{-40}$) that a client may not be able to retrieve all k elements in a single multi-retrieval round.

4.2 A large-scale application of CPIR

In this section, we present a practical application of PIR to ensure privacy in a real-life scenario at Internet scale. The Coeus [3] protocol addresses the issue of private retrieval of a document from a public repository like Wikipedia, based on search keywords.

The problem can be summarized as follows: a user has a search query containing multiple keywords, and a server has a set of public documents. The user inputs the query into a web browser or app, and through interaction with the server, the user can select and view one of the K documents that are most relevant to the query. The privacy requirement is that no one, including the server and potential eavesdroppers, should be able to learn any information about the query or the document the user views.

Coeus uses secure matrix-vector product for document ranking based on the user query using the term frequency-inverse document frequency (tf-idf) statistical method [29] and PIR for document retrieval. The Coeus protocol for privately retrieving a document involves three rounds. In the first round, the user encrypts their query q and sends it to the server. The server responds by sending encrypted relevance scores for the documents. This is achieved by securely multiplying the query vector with the tf-idf matrix. In the second round, the user retrieves short metadata descriptions for the top-K scoring documents from a metadata library. Multi-retrieval PIR [5] (§4.1) is used for this purpose. In the third and final round, the user retrieves a single document that they wish to view in detail. This is accomplished through single-retrieval PIR.

5 EXTENSION OF PIR: KEY-VALUE RETRIEVAL

So far, we focused on situations where the server considers the data as an array of elements and the client knows the array-index of the desired element. However, this requirement of knowing the array-index poses limitations in many practical cases. For example, the server may store the data as a key-value store, and the client may want information associated with a particular key privately, but lacks knowledge of the exact data organization at the server. As an illustration, a customer may want details about a specific stock ticker from a stockbroker or information about a particular disease from a medical repository while keeping the query keywords hidden to safeguard their financial and medical privacy. Due to the dynamic nature of key-value stores, it may not be feasible for the client to know the size of the array or index of a particular element at a given time. In this section, we will explore the challenge of searching over a public key-value store while maintaining privacy.

5.1 Keyword PIR

Chor et al. [13] propose a protocol named *keyword-PIR* to retrieve the value corresponding to a key using multiple sequential rounds of PIR. In this protocol, the server arranges the keys in a searchable data structure, such as a binary search tree, and the protocol progresses in two phases. During the first phase, the client performs PIR on each level of the binary search tree to determine the index of the key. This results in $\lceil \log_2(n+1) \rceil$ round-trip PIR interactions with the server, where n represents the total number of keys. During the second phase, the client utilizes this index to obtain the corresponding value via another round of PIR. However, the drawback of this protocol is that the number of round trip interactions increases with the size of the key-value store, making it a scalability bottleneck. Additionally, since it is a multi-round protocol, it may lead to consistency issues. Ideally, a single round solution for private key-value retrieval would be preferred.

5.2 Single round value retrieval

Conceptually, a single-round solution to the private key-value retrieval problem can be designed using Fully Homomorphic Encryption (FHE) [9, 18] (§3). A client first encrypts the desired key using FHE and sends it to the server. The server obliviously checks equality between the encrypted client key and each key in its key-value store using a homomorphic equality operator. For each key in the key-value store, this step outputs an encryption of 1 if it is equal to the client's query key, or an encryption of 0 otherwise. The output of this step is used to retrieve the desired value using PIR.

The key challenge to the above mentioned approach is to determine equality between two ciphertexts homomorphically. In this tutorial, we briefly discuss two most recent works in this area: Constant-weight Keyword PIR (CKP) [25] and Pantheon [2]. CKP proposes a new method to check if a query key matches any of the keys stored in a key-value store. Their approach involves using a homomorphic equality operator which evaluates a homomorphic boolean circuit to compare the two keys. However, this method requires each bit of the keys to be encrypted separately, resulting in a large number of expensive Multiply (§3) operations, which can create a scalability bottleneck. Pantheon, on the other hand, takes a different approach by using number theory and Fermat's little theorem [28] to develop a homomorphic equality operator. This method significantly reduces the number of ciphertext multiplications required compared to CKP, resulting in much lower computation. For example, when retrieving a value from a key-value store containing 2 million tuples, the latency is less than one second.

6 CONCLUSION AND FUTURE RESEARCH

Private Information Retrieval (PIR) has the potential to enhance privacy in public databases, but further research is necessary to improve its performance and scalability. One active area of research is offline-online PIR [7, 10]. The main idea here is to generate auxiliary information or hints during an offline preprocessing phase of the database and then utilizing these hints during the online phase to answer queries in sublinear time, thereby amortizing the offline phase cost across multiple queries. The field of PIR can be extended to support more expressive queries, such as SQL [27] and graph queries [23], creating another potential area of research. Combining these research areas with cryptographic innovations [20] and the design of specialized hardware [24] for PIR may eventually minimize the cost overhead for privacy.

7 PRESENTERS

Ishtiyaque Ahmad is a Computer Science PhD candidate at the University of California Santa Barbara. His research interest includes preserving privacy in large-scale data management systems. He has published multiple recent papers on developing large-scale practical applications using Private Information Retrieval.

Divyakant Agrawal is a Distinguished Professor of Computer Science at the University of California at Santa Barbara. His research expertise is in the areas of databases, distributed systems, cloud computing, and big data infrastructures and analysis. He served as a Trustee on the VLDB Endowment and is currently serving as the Chair of ACM Special Interest Group on Management of Data (ACM SIGMOD). He is an ACM, IEEE and AAAS Fellow.

Amr El Abbadi is a Professor of Computer Science at the University of California, Santa Barbara. His research interests are in fault-tolerant distributed systems and databases, focusing recently on Cloud data management and privacy preserving data management. He has served as a journal editor for several database journals and has been Program Chair for multiple database and distributed systems conferences. He is an ACM, IEEE and AAAS Fellow.

Trinabh Gupta is an Assistant Professor in the Department of Computer Science at the University of California Santa Barbara. His research interests are in security and privacy properties of computer systems. He has published numerous papers in this area in top venues such as SOSP, OSDI, NSDI, VLDB, and ISCA. He also contributes to this area through industrial activities, especially technology transfer from research to industry.

ACKNOWLEDGMENTS

This work is funded in part by the National Science Foundation under grant number 2126327.

REFERENCES

[1] Carlos Aguilar-Melchor, Joris Barrier, Laurent Fousse, and Marc-Olivier Killijian. 2016. XPIR: Private Information Retrieval for Everyone. In *Privacy Enhancing Technologies Symposium (PETS)*. 155–174.

[2] Ishtiyaque Ahmad, Divyakant Agrawal, Amr El Abbadi, and Trinabh Gupta. 2022. Pantheon: Private Retrieval from Public Key-Value Store. *International Conference on Very Large Data Bases (VLDB)* 16, 4 (2022), 643–656.

[3] Ishtiyaque Ahmad, Laboni Sarker, Divyakant Agrawal, Amr El Abbadi, and Trinabh Gupta. 2021. Coeus: A System for Oblivious Document Ranking and Retrieval. In *ACM Symposium on Operating Systems Principles (SOSP)*. 672–690.

[4] Ishtiyaque Ahmad, Yuntian Yang, Divyakant Agrawal, Amr El Abbadi, and Trinabh Gupta. 2021. Addra: Metadata-private voice communication over fully untrusted infrastructure. In *USENIX Symposium on Operating Systems Design and Implementation (OSDI)*. 313–329.

[5] Sebastian Angel, Hao Chen, Kim Laine, and Srinath Setty. 2018. PIR with compressed queries and amortized query processing. In *IEEE Symposium on Security and Privacy (S&P)*. IEEE, 962–979.

[6] Sebastian Angel and Srinath Setty. 2016. Unobservable communication over fully untrusted infrastructure. In *USENIX Symposium on Operating Systems Design and Implementation (OSDI)*. 551–569.

[7] Amos Beimel, Yuval Ishai, and Tal Malkin. 2000. Reducing the servers computation in private information retrieval: PIR with preprocessing. In *Advances in Cryptology—CRYPTO*. Springer, 55–73.

[8] Zvika Brakerski. 2012. Fully homomorphic encryption without modulus switching from classical GapSVP. In *Advances in Cryptology—CRYPTO*. Springer, 868–886.

[9] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. 2014. (Leveled) fully homomorphic encryption without bootstrapping. *ACM Transactions on Computation Theory (TOCT)* 6, 3 (2014), 1–36.

[10] Ran Canetti, Justin Holmgren, and Silas Richelson. 2017. Towards doubly efficient private information retrieval. In *Theory of Cryptography Conference (TCC)*. Springer, 694–726.

[11] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. 2017. Homomorphic encryption for arithmetic of approximate numbers. In *International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT)*. Springer, 409–437.

[12] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachene. 2016. Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In *International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT)*. Springer, 3–33.

[13] Benny Chor, Niv Gilboa, and Moni Naor. 1998. Private information retrieval by keywords. Cryptology ePrint Archive, Report 1998/003. https://eprint.iacr.org/1998/003.

[14] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. 1995. Private Information Retrieval. In *Symposium on Foundations of Computer Science (FOCS)*. 41–50.

[15] Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. 1998. Private information retrieval. *Journal of the ACM (JACM)* 45, 6 (1998), 965–981.

[16] Taher ElGamal. 1985. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE transactions on information theory* 31, 4 (1985), 469–472.

[17] Junfeng Fan and Frederik Vercauteren. 2012. Somewhat practical fully homomorphic encryption. *Cryptology ePrint Archive* (2012). https://eprint.iacr.org/2012/144.pdf

[18] Craig Gentry. 2009. Fully homomorphic encryption using ideal lattices.. In *ACM Symposium on Theory of Computing (STOC)*. 169–178.

[19] Trinabh Gupta, Natacha Crooks, Whitney Mulhern, Srinath Setty, Lorenzo Alvisi, and Michael Walfish. 2016. Scalable and private media consumption with Popcorn. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*. 91–107.

[20] Alexandra Henzinger, Matthew M Hong, Henry Corrigan-Gibbs, Sarah Meiklejohn, and Vinod Vaikuntanathan. 2023. One Server for the Price of Two: Simple and Fast Single-Server Private Information Retrieval. In *USENIX Security Symposium (SEC)*.

[21] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. 2004. Batch codes and their applications. In *ACM symposium on Theory of computing*. 262–271.

[22] Eyal Kushilevitz and Rafail Ostrovsky. 1997. Replication is not needed: Single database, computationally-private information retrieval. In *Symposium on Foundations of Computer Science (FOCS)*. 364–373.

[23] Ling Liang, Jilan Lin, Zheng Qu, Ishtiyaque Ahmad, Fengbin Tu, Trinabh Gupta, Yufei Ding, and Yuan Xie. 2023. SPG: Structure-Private Graph Database via SqueezePIR. *International Conference on Very Large Data Bases (VLDB)* 16, 7 (2023), 1615–1628.

[24] Jilan Lin, Ling Liang, Zheng Qu, Ishtiyaque Ahmad, Liu Liu, Fengbin Tu, Trinabh Gupta, Yufei Ding, and Yuan Xie. 2022. INSPIRE: In-Storage Private Information Retrieval via Protocol and Architecture Co-design. In *International Conference on Computer Architecture (ISCA)*. 102–115.

[25] Rasoul Akhavan Mahdavi and Florian Kerschbaum. 2022. Constant-weight PIR: Single-round Keyword PIR via Constant-weight Equality Operators. In *USENIX Security Symposium (SEC)*.

[26] Pascal Paillier. 1999. Public-key cryptosystems based on composite degree residuosity classes. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*. Springer, 223–238.

[27] Xuanle Ren, Le Su, Zhen Gu, Sheng Wang, Feifei Li, Yuan Xie, Song Bian, Chao Li, and Fan Zhang. 2022. HEDA: Multi-Attribute Unbounded Aggregation over Homomorphically Encrypted Database. *International Conference on Very Large Data Bases (VLDB)* 16, 4 (2022), 601–614.

[28] Kenneth H Rosen. 2011. *Elementary number theory*.

[29] Gerard Salton and Michael J. McGill. 1986. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc.

[30] Julien P Stern. 1998. A new and efficient all-or-nothing disclosure of secrets protocol. In *International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT)*. Springer, 357–371.

[31] Rei Yoshida, Yang Cui, Rie Shigetomi, and Hideki Imai. 2008. The practicality of the keyword search using PIR. In *2008 International Symposium on Information Theory and Its Applications*. IEEE, 1–6.