Remaining Private in the World of Great Data Exchange

William Kovacs
Stanford University

Abstract

With an ever increasing number of databases that people access on a daily basis, user privacy becomes an increasing issue. In order to ensure that not even the owners of the database can determine the records that a particular user is trying to access, plenty of research has been devoted to developing Private Information Retrieval schemes capable of masking a user's queries. Such schemes can be broken into two categories: the information-theoretic and the computationally-bounded approaches. Both styles will be examined, along with a quick look at how they can be combined to improve performance.

1 Introduction

From watching movies on Netflix to watching NASDAQ's stocks rise and fall to watching looming love interests on eHarmony, databases have become an essential aspect of many people's lives. As such, people can feel that their privacy is threatened. Some modern techniques, such as user-targeted ads, can further exacerbate this feeling of diminished privacy. After all, if the site knows what you are searching, what's preventing a curious database manager from examining any user's queries?

In order to ensure that a user's queries remains private, a Private Information Retrieval (PIR) system can be used to mask them. Essentially, when a user accesses a database via PIR, the query and response are performed in such a way that the owner of the database has no knowledge of which record was retrieved. It should be stressed that a PIR system only protects the privacy of the user, and not of the database. As such this type of system can be thought of as a weaker form of oblivious transfer. Indeed, in some circumstances, it is possile to create an oblivious transfer scheme from a PIR scheme [6].

A trivial, though incredibly inefficient, solution to PIR is to send the entire database to the user. Instead, the main goal of PIR research is to provide a retrieval system that is faster than this trivial method, while retaining the same privacy. Currently, there are two main approaches in this field: the information theoretic approach, which relies on the use of multiple servers, and computational PIR (cPIR), which relies on the computational limitations of a single server. In prior years, the former method has been preferred due to the computationally expensive nature of cPIRs. Indeed, in their oft-cited paper, Sion and Carbunar posited that cPIR protocols were simply not practical [19]. While this may have been true at the time, research since then have come to refute such claims and cPIR has become a feasible alternative to systems requiring multiple servers [16]. Through the remaining sections of this paper, exemplars of the different model types will be examined.

2 Informational Theoretic PIR

Chor et al. were the first to introduce the PIR problem, and provide a solution that utilizes distributed databases [5]. In order to be secure, the servers housing the databases must not collude with each other. Chor et al. build their solution one piece at a time, starting from a simple, two-server scheme that is capable of privately retreiving a single bit. This scheme is then improved to utilize multiple servers in a more efficient manner, before finally being extended to blocks of data.

2.1 Two-Server Bit Retrieval

Say a user is trying to identify the i'th bit of an n-bit database, copies of which are located on two separate servers. The user can generate a random subset of indices (containing values from 1 to n, each with a probability of 1/2 of being included), followed by the creation of a secondary subset, which contains all of the same elements as the previous one, except for the i'th one. That is, i is present in only one of the two subsets, and the rest of the indices are shared between the two. The user sends one subset to the first server, and the other subset to the second. The servers then extract the bit values present at the provided indices, XORs them together, and returns those values to the user. The user can then XOR the two bits that he receives to get the value of the bit at the desired index.

The correctness of this scheme can be seen by understanding that the values retrieved by the servers are from all of the same indices except for the i'th one, so in the XORs, these values cancel each other out, leaving only the value of i. Furthermore, because each subset appears random to each server, the servers gain no information about the desired index. However, this scheme does not improve on the communication costs compared to the trivial method because the user still has to send n-bit strings to the servers instead of receiving one. Instead, the underlying idea can be adapted to increase efficieny, as discussed in the next section.

2.2 Multi-Server Bit Retrieval

By using 2^d servers and embedding the indices in a d-dimensional cube, the communication costs of the PIR scheme can be reduced, due to the succint representation of 'subcubes' as compared to the afore-used subsets. The length of each side of the cube is $\sqrt[d]{n}$, and each index is associated with a tuple corresponding to the coordinates of a unique location in the cube. Then, the two server scheme to generate subsets is applied along each dimension. That is, for each dimension, a random subset of indices is chosen, and a secondary subset is generated where the coordinate for the i'th index differs, generating d pairs of subsets.

One of each pair of subsets is sent to each server. To decide which pair, the servers are labelled with bit-strings of length d. For the subset corresponding to dimension k, if the k'th bit of the label is a 0, the original random subset is sent, and if it's a 1, the switched one is sent. This ensures that each server receives a unique combination of subsets. The server then XORs the values at the indices corresponding to the subcube (that is all of the combinations of the coordinates) defined by the subsets, and sends that back to the server. The user then XORs the values to retrieve the desired bit. This works because the coordinate corresponding to the i'th index only appears in one subcube (this can be intuitively seen by knowing that for each pair of subsets, only one contains the coordinate corresponding to i, and each of these correct subsets needs to be sent at the same time, which only happens once), while the remaining ones appear an even number of times, so the XORing cancels all but the i'th index. The communication cost for this scheme is $2^d \cdot (d \cdot \sqrt[d]{n} + 1)$, because for each of the 2^d servers, the user sends $\sqrt[d]{n}$ bits and receives 1.

To reduce the cost of the multi-server scheme, the authors propose that a single server can emulate a server whose bit-label has a Hamming distance of 1 (as those further apart would require too many extra computations). For the single dimension that the two servers would disagree on, the emulating server knows that a single member of the subset may be switched (that is included or excluded), and so the server can send back the $\sqrt[d]{n}$ bits corresponding to queries where each index along the differing dimension is changed. The user then XORs the results and retrieves the desired bit for similar reasons as described in the vanilla multi-server scheme. This change decreases the cost meaningfully at lower values of d, as it depends on the minimum number of servers that can emulate other servers without overlapping.

2.3 Block Retrieval

While developing schemes for single bit PIR demonstrated its feasibility, Chor et al. wanted to also generate a practical model, so they described how to transform the single-bit retrieval models into a single-block retrieval scheme, where each block can represent a record. A simple transformation that they suggest begins by organizing the data into an $m \cdot n$ matrix, where each column represents a block of data. The user can then query for the i'th bit of a row using one of the above protocols, and the server returns the results for this query for each row of data. This gives the user the column of the matrix corresponding to the desired block of data. This results in an asymmetric communication cost, that is as the size of the blocks increases, the cost from the server to the user increases, while the reverse direction remains the same.

Overall, Chor et al. provided a robust, initial exploration for utilizing multiple servers to create a PIR system, and laid the foundation for information theoretic PIR.

2.4 cPIR

The goal of cPIR differs from the above, multi-server scheme in that instead of providing information theoretic secrecy, it provides privacy by making it computationally infeasible for a single server to identify the user's query, usually through the use of cryptography. Here, we will examine one style of cPIR that relies on the use of homomorphic encryption, as it provides an avenue for a cPIR system that is capable of running faster than the trivial PIR. The privacy of the following methods can be easily seen because any values that the database sees are encrypted, so all elements are handled equally, in the database's view.

2.5 Homomorphic Encryption PIR

Before focusing on a specific cPIR system, it is helpful to understand the generalized case of a cPIR system that relies on homomorphic encryption. This type of encryption scheme allows for operations to be performed on a ciphertext, such that the operations will also be performed on the underlying message. This message space will be considered as the group G. For simplicity, it is easier to discuss G as an additive group, so we can consider the operation of the group as a sum, the Z-module action as a multiplication, and the identity element as 0. Ostrovsky and Smith [17] present a series of general versions of PIR systems where any homomorphic cryptoscheme can be inserted. These protcols share many similarities to the multi-server schemes presented by Chor et al.

For instance, the initial PIR scheme resembles the two server scheme described in section 2.1. The user sends n queries to the server, where n is the number of elements in the server, and each each i'th query corresponds to the i'th index of the database. The queries are encrypted messages using the selected cryptoscheme, where the query corresponding to the desired index is an encryption of a random element, g, in G, and all other queries are encryptions of 0.

For each query, the database can multiply the index bit with that query, resulting in a ciphertext of 0 if the bit is not the desired one. If the query corresponds to the desired bit, the resulting ciphertext will either be g or 0 depending on the value of that bit. The server can then sum all of the queries, which results in the ciphertext of g only if the desired bit is 1. The database can then send this back to the user, who can then decrypt it and compare it against g. This protocol suffers from the same issue as the aforementioned two-server scheme: it's communication cost is the same as the trivial PIR method because it requires sending n queries to the server.

2.5.1 2-D Database cPIR

In order to improve this cost, a similar extension to the one proposed in Section 2.2 can be used: instead of viewing the database as one long, bit string, it can be imagined as a square of length $\sqrt{n}$, where the desired index now resides at some coordinate, (i, j). To retrieve this bit, the user can perform the query used above along one of the dimensions. For instance, for each row of the square, the user sends a query that is again either an encryption of 0 if the row that is being queried does not contain the desired bit, or the encryption of some arbitrary element, g, if the row does contain the desired element. For each column of the database, the server can generate a response using the protocol described (i.e. multiplying each index bit with the query, and summing these together), and send back the response to the user, resulting in $\sqrt{n}$ responses back, one for each column. The user can then decrypt the response corresponding to the desired column, and check whether or not the message equals g. If so, then the user knows the desired bit is 1, otherwise it is 0. The communication cost for this scheme now becomes $O(\sqrt{n})$ because $\sqrt{n}$ queries are both sent to and received from the server.

2.5.2 Alternate 2-D Database cPIR

An improvement can be made on this scheme that holds similar communication costs for the 2-D representation, but allows for an externsion for greater communication efficiency via a d-dimensional represenatation. This method requires the use of an injective map that can take some cipher text, and map it to a an array of length l, whose elements are less than the order of g. l is always greater than 1 because the order of g is always less than the size of the message space, which is always less than the size of the ciphertext space because the encryption scheme is probabilistic. If l was 1, then there is a guaranteed duplicate mapping via the pigeonhole principle.

To start this protocol, instead of sending queries for only one dimension, queries along both dimensions are sent, with the same property as above: an encryption of g if the desired point is along that dimension, otherwise, of 0. This generates $\sqrt{n}$ queries for each dimension. The server then processes the query in a similar fashion as above, where each of the elements are multiplied with the corresponding query element, and are summed together along one dimension, resulting in $\sqrt{n}$ elements. Instead of sending these back to the user, the injective map is applied to each element, producing an $l \times \sqrt{n}$ matrix. This can be thought of as a new database, and the same procedure can be applied along the remaining $\sqrt{n}$ dimension using the relevant queries, ending up with l elements. Each of these corresponds to g times one of the l elements of the result of the mapping.

These elements are sent back to the user, who can easily reconsruct the injective mapping. The inverse mapping can be applied to retrieve the value that would have been sent back in the scheme in Section 2.5.1. The user then needs to check if the value eqauls g or not to determine the desired bit. The communication cost is now $(2 * \sqrt{n} + l) * k$, where k is the length of the queries. This can be seen because the user has to make $\sqrt{n}$ queries for each dimension, and the database responds with the l results of the corresponding injective map.

2.5.3 d-D Database cPIR

With the use of these injective maps, the method can be extended such that the database can be organized as a d-dimensional cube with length $\sqrt[d]{n}$. This is also referred to as a recursive method. The idea here is that there are a series of $\sqrt[d]{n}$ queries for each dimension. When the method described in Section 2.5.1 is used for one of the sets of queries, the length of the cube along the corresponding dimension is reduced to 1. The remaining elements correspond to the bits along the remaining dimensions times the encrypted value of g or 0, depending on whether the elements were along the desired dimension. The use of the injective map as described in Section 2.5.2 inflates the length of this dimension up to l, masking these values, and creating a new 'database' with the same number of dimensions. Then, the same procedure can be applied to this new database along a different dimension.

After all sets of queries have been applied, the server is left with a $d-1$-dimensional cube with length l that it sends back to the user. The cube only has $d-1$ dimensions because the injective map is not applied along the last dimension. Each element of this returned cube can be thought of as a resulting segment of the mapping of a resulting segment of a mapping, etc, of the desired record. Then, to retrieve the desired record, the user has to invert the mapping along each dimension in the reverse order that the queries were applied (the last applied injective map needs to be inverted before any of the other ones). Each mapping was multiplied by the value g, as the hidden record corresponds to the desired index whose queries all correspond with g, so the user has to also remove this contribution before applying the inverse. The user is then left with the desired record.

The communication cost of this protocol is now $O(kd\sqrt[d]{n}+l^{d-1})$ because the user sends $d\sqrt[d]{n}$ of length k queries to the databse, which returns the the $(d-1)$-dimensional cube with sides of length l.

In the following described system, this recursive scheme is the one that is used.

2.6 XPIR

In PIR, the major goal is to reduce the communication costs to below that of the trivial solution. However, to make these methods practical, it is also important to ensure that the computational costs do not outweigh the communication costs, as the major issue in cPIR for a long time was that the computational overhead outweighed the communication benefits. In this section, I will focus on Aguilar-Melchor et al.'s paper, wherein they use the cPIR scheme described in Section 2.5.3, with a homomorphic encryption scheme based on the ring learning with error (RLWE) problem [3, 11]. Furthermore, they also try to reduce computation costs by using an efficient Number Theoretic Transform (NTT), Chinese Remainder Theorem (CRT) repesentation, and precomputation of costly operations.

2.6.1 The Encryption Scheme

The basis for security in the XPIR scheme is due to the ring learning with errors (RLWE) problem. For the purpose of this survey, only a high-level overview of this problem is presented, structured as a game. For further details, please consult [11]. To begin with, the elements of this scheme come from a ring of polynomials, R_q, whose coefficients are all relative integers modulo q, and any operation performed between polynomials is done so modulo x^N+1, where N is the degree of the polynomial.

The RLWE problem begins by the challenger selecting a random polynomial, s, and a random 'small' polynomial, e, from this ring. For simplicity, e, which will represent the 'error', is considered small when it has small integer coefficients. Then the challenger can release one of two sets of outputs: 1) for each output, a random polynomial, a, is selected from the ring and the operation $b = a\cdot s+e$ is performed, and (a,b) is the output. 2) for each output, two random polynomials, a and r are selected from the ring, and (a,r) is output. The adversary then has to decide which of the two sets of outputs he receives. The indistinguishability of these two outputs, and thus its security, is based on the hardness of the shortest vector problem on ideal lattices.

A public-key encryption scheme that utilizes the hardness of this problem is described in [3], which XPIR uses in its own implementation, and is briefly described as follows. It should be noted that when all the e polynomials are generated, their coefficients are scaled by t, the maximal coefficient of the message. The secret key, sk, is simply a random polynomial in R_q. The public key consists of two polynomials: pk_1 is another random polynomial in R_q, and $pk_2 = pk_1 * sk + e$. To encrypt a message, a u and an e are randomly selected from the ring, as well as an e', whose randomness has a larger variance than the e for pk_1. The ciphertext consists of two parts: an $a = pk1 * u + e$ and a $b = pk2 * u + e' + m$. To decrypt, simply compute $b-(a*sk) \bmod t$. As this paper focuses on PIR, the analysis of this scheme is left to [3]. Essentially, if Ring-LWE is hard, then this scheme is as well.

2.6.2 The PIR scheme

The cPIR scheme used in this system is the same as that described in 2.5.3, where the binary representation of the encryption algorithm takes place of the injective maps. However, the system has to be adapted to retrieve blocks of data instead of just single bits. Furthermore, with this RLWE based encryption scheme, only l_0 bits can be stored in a ciphertext at one time, otherwise it is impossible to decrypt correctly [3]. l_0 is chosen such that the ratio between q and the elements of e remains greater than 2. These conditions requires changing only the reply from the server, and the way that the user interprets the reply.

For the basic case where the server is a linear series of records, each record would be split up into blocks of size l_0, resulting in l/l_0 blocks, where l is the length of a record. Then the aforementioned scheme is applied for each corresponding block: the block is multiplied with the query, and the corresponding blocks of each record (e.g. all the first blocks) are summed together, resulting in l/l_0 blocks that are sent back to the user. The user can then retrieve the message by decrypting the blocks and concatenating them together. This change can then be easily extended into the cases where the server is structured as a d-dimensional cube by replacing each response with this series of blocks.

2.6.3 Optimizations

In order to make the above encryption scheme more efficient, XPIR represents the polynomials using NTT, and represents integers using CRT, similar to [9]. NTT can be thought of as the Fast Fourier Transform applied to polynomials that are members of the ring [1], while CRT allows numbers that are mod large values to be converted to and from a series of numbers mod small primes, allowing for quicker computations.

To improve the efficiency of costly modular, multiplications, they use an algorithm attributed to Shoup, and used in NTL [10, 18]. The basic idea is to precompute the costly division step when the multiplicand is used many times, that is for a given xy mod p, where x is going to remain the same for many such computations, $x' = (x * \beta)/p$ is precomputed. To get this division to work appropriately, it occurs in 128 bits (whereas the integers themselves are stored in 64 bits), and β is a scaling factor equal to 2^{64}. During runtime, the quotient of the division for the modular arithmetic is approximated by computing $Q = x'y/\beta$. This value is shown to be either the correct quotient or too large by 1 in [10]. To calculate the modular remainder, $xy - Qp$ mod β is calculated, and if this is greater than p, p is subtracted from this value.

This algorithm improves the efficiency of these calculations because it moves the costly division step to a precomputation step, so during runtime, all that is needed is two integer multiplications, a shift, and a condiftional subtraction. Furthermore, because many of the multiplications performed using this algorithm have an invariant factor, that is the secret key, public key, or the query elemnts, this method allows for substantial speed up.

3 Performance of PIR systems

The main purpose of XPIR was to demonstrate that a feasible cPIR system could be constructed through the use of the model and optimizations described. To demonstrate this, Aguilar-Melchor et al. tested data throughput when using xPIR against two different types of databases: static databases whose data could be preprocessed, and dynamic ones whose data could not. When the database was viewed as one dimensional, their scheme was 100x faster in the former case, and 50x faster in the latter. When the database was viewed as a square, the throughput efficiency dropped by half for small databases, due to the overhead of the intermediate 'database'. However, as the size of the database grows, the difference between throughput of the two views decreased, until they converged when the databse consisted of 10,000 records.

While the reorganization of the database does present some throughput limitations on smaller databases, it's main advantage is that the user can start receiving data quicker after she sends a query, because the communication cost is lower. Indeed, when the database only contained 10 files, there was no noticeable difference in such latency. However, when the database grew to only 100 files, the user would receive the initial response 10x quicker with the reorganization (0.1s vs 1s when a fiber connection was simulated). The gap changed to 100x quicker (1s vs 100s) with a 2-D database when it consisted of 10000 files.

While this does demonstrate the feasibility of these methods , it should still be noted that the multi-server schemes still demonstrate better performance over these cPIR systems, as to be expected. In a comparison of these two types of schemes, Olumofin and Goldberg demosntrated that Chor's basic PIR scheme (Section 2.3) had a 100x improvement of reponse times over the cPIR scheme that was used [13]. However, this comparison used an older version of a lattice-based cPIR and was performed on older hardware, so direct comparison to the xPIR scheme is difficult. Furthermore, the multi-server scheme used in this comparison was itself not optimal, and its performance could be improved by 50x by choosing an appropriate block size for the system based on empirical data as opposed to the communication costs theory [4]. Though, the main point still remains that while both types of schemes offer performance better than the trivial case, the multi-server schemes do not suffer from computational drawbacks as severely as cPIR ones.

4 Hybrid Methods

cPIR methods offer several advantages over multi-server PIR schemes that are gated behind their computational costs, so there have been several attempts to combine these two types of schemes. Devet and Goldberg propose a scheme that does this in order to reduce the overall communication cost by utilizing the efficiency of the recursion schemes of cPIR systems within the already efficient multi-server scheme. They present the scheme in a generalized format for any multi-server PIR and cPIR combination. Like the individual schemes, the database can be represented as a d-dimensional cube, where d is chosen to minimize communication costs. For the first dimension, the multi-server scheme is used. If communication costs are found to be minimum when $d = 1$, then the result is sent back to the user, and the entire scheme functions as a multi-server scheme. This ensures that the set-up performs no worse than the multi-server scheme. However, if d is greater than 1, the cPIR scheme is recursively applied on each server, as described in Section 2.5.3, to the newly generated block.

In their implementation, Devet and Goldnerg used a multi-server scheme based on Shamir-secret sharing [7], as well as a lattice based cPIR system [14], that is now thought to be broken [12]. As these methods are incidental to the overall hybrid scheme, the full description is not demonstrated here. By testing their system against different sized databases, Devet and Goldberg demonstrate that it can be used with costs no worse than the multi-server scheme, and improved performance when records are sufficiently small compared to the size of the database.

Popcorn takes a different approach to a 'hybrid' method [8]. This system is designed to deliver media, such as movies, from a static database, while adhering to content protection policies. Here, the scheme uses the multi-server scheme described in Section 2.3 to effeciently retrieve encrypted movies from secondary databases, but, in order to comply with content policies, it uses XPIR (Section 2.6) to retrieve the keys from a single primary database. Such a scheme is based on current DRM schemes, as it allows efficient distribution of content, while ensuring that only the original distributor retains original copies. By delegating most of the work to the multi-server scheme and only the key distribution to the cPIR scheme, this system was shown to be a possible method to implement PIR for this industry, if the size of the database isn't too large.

5 Conclusion

As our society's reliance on retrieving data from other entitites for work, for pleasure, for life, continues to grow, so too does the concern for privacy. Since it's formalization by Chor et al., the basic PIR scheme has proven to be a powerful starting point to address this issue. It provides a simple solution with its use of multiple servers that has been expanded in order to provide more practical schemes, such as allowing some servers to not be truthful and retaining privacy in cases where some servers may collude [7].

Due to recent advances in cryptography, particularly lattice based schemes, cPIR, once thought to be infeasible, has become a possibility. This allows for the distribution of data from a single server, permitting greater flexibility in setting up PIR schemes, particularly in cases where owners of a database don't want to distribute their content to outside parties. It still remains to be determined whether current systems are efficient enough to be used in the real world, and whether further improvements can be made on the lower bounds of their costs.

Scalability still remains an issue with these schemes. They tend to work well with servers that have small, stable data, which are often compared to a Netflix-like database. Unfortunately , there still has yet to be a PIR system that can scale well to a Youtube-sized one. While such a system would be impressive, it's also possible that using the current schemes to hide the query within a subset of other queries could be sufficient. After all, to some people, there may not be much of a differene between an adversary distinguishing the user's query from all 1,000,000,000 possible queries and being able to narrow the user's query to one of 100,000 queries.

While there have been very exciting trends towards making the simple block retrieval PIR systems more efficient, in order for them to be truly practical, these schemes need to be able to work within the current framework for retrieving records from a database. The aforementioned schemes have all focused on retrieving blocks from a simple index-based database. Currently, relational databases that use SQL to retrieve records in a more flexible manner are a popular choice, and there have been attempts to adapt these schemes to ensure privacy over SQL queries with moderate success [15]. For these types of databases, a recent breakthrough, function secret sharing, allows for the construction of highly efficient schemes [2, 20].

With many resources becoming engrained in a digital setting, PIR may become an important asset to help maintain user privacy. With each further improvement, these schemes get closer to becoming truly practical, and with improved efficiency and less restrictions, it may become a desirable scheme used in modern applications.

References

[1] ARNDT, J. *Number theoretic transforms (NTTs)*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 535–542.

[2] BOYLE, E., GILBOA, N., AND ISHAI, Y. *Function Secret Sharing*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015, pp. 337–367.

[3] BRAKERSKI, Z., AND VAIKUNTANATHAN, V. Fully homomorphic encryption from ring-lwe and security for key dependent messages. In *Proceedings of the 31st Annual Conference on Advances in Cryptology* (Berlin, Heidelberg, 2011), CRYPTO'11, Springer-Verlag, pp. 505–524.

[4] Cappos, J. *Avoiding theoretical optimality to efficiently and privately retrieve security updates*, vol. 7859 LNCS of *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. 2013, pp. 386–394.

[5] Chor, B., Kushilevitz, E., Goldreich, O., and Sudan, M. Private information retrieval. *J. ACM 45*, 6 (Nov. 1998), 965–981.

[6] Di Crescenzo, G., Malkin, T., and Ostrovsky, R. *Single Database Private Information Retrieval Implies Oblivious Transfer*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2000, pp. 122–138.

[7] Goldberg, I. Improving the robustness of private information retrieval. In *2007 IEEE Symposium on Security and Privacy (SP '07)* (May 2007), pp. 131–148.

[8] Gupta, T., Crooks, N., Mulhern, W., Setty, S., Alvisi, L., and Walfish, M. Scalable and private media consumption with popcorn. Cryptology ePrint Archive, Report 2015/489, 2015. `http://eprint.iacr.org/2015/489`.

[9] Halevi, S., and Shoup, V. Design and implementation of a homomorphic-encryption library.

[10] Harvey, D. Faster arithmetic for number-theoretic transforms. *Journal of Symbolic Computation 60* (2014), 113 – 119.

[11] Lyubashevsky, V., Peikert, C., and Regev, O. On ideal lattices and learning with errors over rings. *J. ACM 60*, 6 (Nov. 2013), 43:1–43:35.

[12] Melchor, C. A., Barrier, J., Fousse, L., and Killijian, M. XPIR : Private information retrieval for everyone. *PoPETs 2016*, 2 (2016), 155–174.

[13] Melchor, C. A., Crespin, B., Gaborit, P., Jolivet, V., and Rousseau, P. High-speed private information retrieval computation on gpu. In *2008 Second International Conference on Emerging Security Information, Systems and Technologies* (Aug 2008), pp. 263–272.

[14] Melchor, C. A., Crespin, B., Gaborit, P., Jolivet, V., and Rousseau, P. High-speed private information retrieval computation on gpu. In *2008 Second International Conference on Emerging Security Information, Systems and Technologies* (Aug 2008), pp. 263–272.

[15] Olumofin, F., and Goldberg, I. Privacy-preserving queries over relational databases. In *Proceedings of the 10th International Conference on Privacy Enhancing Technologies* (Berlin, Heidelberg, 2010), PETS'10, Springer-Verlag, pp. 75–92.

[16] Olumofin, F., and Goldberg, I. Revisiting the computational practicality of private information retrieval. In *Proceedings of the 15th International Conference on Financial Cryptography and Data Security* (Berlin, Heidelberg, 2012), FC'11, Springer-Verlag, pp. 158–172.

[17] Ostrovsky, R., and Skeith, W. E. *A Survey of Single-Database Private Information Retrieval: Techniques and Applications*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007, pp. 393–411.

[18] Shoup, V. NTL: A library for doing number theory. http://www.shoup.net/ntl, 2003.

[19] Sion, R., and Carbunar, B. On the practicality of private information retrieval. In *Proceedings of the Network and Distributed System Security Symposium, NDSS 2007, San Diego, California, USA, 28th February - 2nd March 2007* (2007), The Internet Society.

[20] Wang, F., Yun, C., Goldwasser, S., Vaikuntanathan, V., and Zaharia, M. Splinter: Practical private queries on public data. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)* (Boston, MA, 2017), USENIX Association, pp. 299–313.