Hypercubes and Private Information Retrieval

Anirban Basu(✉), Rui Xu, Juan Camilo Corena, and Shinsaku Kiyomoto

KDDI Research, Fujimino, Japan
{basu,ru-xu,corena,kiyomoto}@kddi-research.jp

Abstract. In geometry, a hypercube is a regular polytype – a generalisation of a 3-dimensional cube to λ-dimensions, with mutually perpendicular sides of equal lengths. For $\lambda = 0, 1, 2, 3,$ and 4, a hypercube is a point, a straight line segment, a square, a cube and a tesseract respectively. In this paper, we apply the concept of hypercubes in computationally private information retrieval (CPIR) based on additively homomorphic cryptosystems and optimise it further at the cost of a measurable privacy loss.

Keywords: Privacy · Hypercubes · Information retrieval

1 Introduction

A *computationally private information retrieval* (CPIR) lets a receiver retrieve an l-bit element from the sender's database of n elements without revealing the retrieved element to the sender. This is a weaker version of the 1-out-of-n oblivious transfer, which ensures that the receiver is unable to obtain any information about the other elements in the sender's database.

CPIR is useful in many real world scenarios. For example, in an opinion poll, the identity of the person submitting the opinion should be decoupled from the opinion itself to facilitate unbiased polls, and yet ensure that only a set of authorised entities are allowed to submit the opinions. One way of doing this is to let every authorised person pick a valid token using CPIR, and then submit opinions where every opinion is tied to a previously picked valid token. Even if the identity of the submitter is not concealed (unless using anonymous networking) the submitter can plausibly deny that the submitted opinion is hers because the poll administrator cannot prove, due to CPIR, that a particular token was picked by her.

In geometry, a hypercube is a generalisation of a 3-dimensional cube to λ-dimensions, with mutually perpendicular sides of equal length, d.

J. C. Corena—Portions of this work are contributions from Juan Camilo Corena when he was at KDDI Research (erstwhile KDDI R&D Laboratories). He currently works for Google. This work is not related to or supported by Google in any way. He is also reachable at investigacion@juancamilocorena.com.

S. Qing et al. (Eds.): ICICS 2017, LNCS 10631, pp. 509–515, 2018.
https://doi.org/10.1007/978-3-319-89500-0_44

For $\lambda = 0, 1, 2, 3,$ and 4, a hypercube is a point, a straight line segment, a square, a cube and a tesseract respectively.

In this paper, we describe how the concept of hypercubes could be utilised in computationally private information retrieval (CPIR), which is very similar to the scheme described by Chan [1]. We propose a method to improve the performance of the hypercube-backed CPIR at the cost of a measurable loss in privacy.

The rest of the paper is structured as follows. We present a brief overview of the state-of-the-art in private information retrieval in Sect. 2. This is followed by some background in homomorphic encryption in Sect. 3 before we delve into describing our CPIR protocol based on λ-dimensional hypercubes in Sect. 4 with an optimised version in Sect. 4.2. We present technical feasibility through evaluations of cryptographic primitives in Sect. 5 before concluding in Sect. 6.

2 Related Work

The problem of hiding the index of a retrieval operation on a database from the server which actually holds the database was investigated by Rivest et al. [2], Blakely and Meadows [3], Abadi et al. [4,5], Beaver and Feigenbaum [6]. The current known seminar work on private information retrieval by Chor et al. [7,8] builds upon the above. PIR can be roughly grouped into two categories, information-theoretic PIR and computationally PIR. The initial proposals of Chor et al. [7,8] assume k non-communicating servers to store the database and can resist computationally unbounded malicious servers. Later on the weaker notion of computational PIR [9], which aims only at providing privacy against computationally bounded adversary, emerges so as to relieve the critical assumption on more than one non-communicating servers.

The work very close to our scheme is by Chan [1], which uses 2-D hypercube and its generalisations into higher dimensions for a single server private information retrieval with $\mathcal{O}(n)$ communication complexity. The work is more computationally efficient on the server side than ours but has more computations (than ours) on the client-side. Chan's work uses the Damgärd-Jurik cryptosystem, reducing the need for a larger homomorphic cryptosystems for every hypercube dimension reduction than the previous reduction. The main difference between our work and [1] is that we propose a version of PIR, in which we can reduce a lot of the computational complexity at the cost of a measurable privacy loss, which is explained in Sect. 4.2.

3 Background – Homomorphic Encryption

Homomorphic encryption allows computing over encrypted data without requiring the knowledge of either the actual data or any results produced through the computation. Depending on the type of computational operations supported, homomorphic cryptosystems are classified as: (1) additive, (2) multiplicative,

(3) somewhat homomorphic (e.g., allowing a number of additions and one multiplication), and (4) fully homomorphic.

The Paillier public-key cryptosystem [10], satisfying semantic security against chosen plaintext attacks (IND-CPA requirement), and its variant, the Damgård-Jurik cryptosystem [11], have practical implementations and both exhibit only additively homomorphic properties: (a) the encryption of the sum of two plaintext messages m_1 and m_2 is the modular product of their individual ciphertexts, i.e., $\mathcal{E}(m_1 + m_2) = \mathcal{E}(m_1) \cdot \mathcal{E}(m_2)$ and (b) the encryption of the product of one plaintext message m_1 and another plaintext multiplicand π is the modular exponentiation of the ciphertext of m_1 with π as the exponent, i.e., $\mathcal{E}(m_1 \cdot \pi) = \mathcal{E}(m_1)^{\pi}$.

4 Computationally Private Information Retrieval (CPIR) Using λ-Dimensional Hypercubes

In private information retrieval, given a database of elements $T = t_1 t_2 t_3 \dots t_n$, the aim is to retrieve a t_x such that the database owner cannot learn which t_x was retrieved. Since n can be very large, T is folded into a λ-dimensional hypercube. This means that each edge of the hypercube will contain $d = \sqrt[\lambda]{n}$ elements. Finding a t_x is essentially locating a point on the λ-dimensional coordinate space.

Essentially, the responder sends λ vectors of encrypted 0s and encrypted 1s, each of length $d = \sqrt[\lambda]{n}$ and each having exactly one encrypted 1 while the rest are encrypted 0s. Each encrypted vector multiplied with a multi-dimensional hypercube helps reducing the hypercube by one dimension until it reduces to a single point. This process, depending on the way it is done, may require fully homomorphic encryption.

Let us see how this works for $\lambda = 2$ i.e., a 2-D hypercube or a square matrix. Assume that all the tokens in T are arranged in the square matrix of size $d \times d$, as:

$$T = \begin{pmatrix} t_{1,1} & t_{1,2} & t_{1,3} & \dots & t_{1,d} \\ t_{2,1} & t_{2,2} & t_{2,3} & \dots & t_{2,d} \\ t_{3,1} & t_{3,2} & t_{3,3} & \dots & t_{3,d} \\ \dots & \dots & \dots & \dots & \dots \\ t_{d,1} & t_{d,2} & t_{d,3} & \dots & t_{d,d} \end{pmatrix} \tag{1}$$

We can reduce this with two encrypted vectors of zeros and ones, each of size n: $V_a = \{\mathcal{E}_a(v_{a,1}), \mathcal{E}_a(v_{a,2}), \mathcal{E}_a(v_{a,3}), \dots, \mathcal{E}_a(v_{a,m})\}$ and $V_b = \{\mathcal{E}_b(v_{b,1}), \mathcal{E}_b(v_{b,2}), \mathcal{E}_b(v_{b,3}), \dots, \mathcal{E}_b(v_{b,d})\}$ where any of $v_{a,k}$ or $v_{b,k}$ is either a zero or one. Exactly one component in each vector is an encrypted one. The encryption function $\mathcal{E}_b()$ is such that its plaintext space is same as or more than the ciphertext space of $\mathcal{E}_a()$, for example $\mathcal{E}_b()$ could be a 512 bits Paillier cryptosystem while $\mathcal{E}_a()$ is a 256 bits Paillier cryptosystem.

Homomorphically multiplying the first row of T with V_a and homomorphically summing the components, we will produce: $\mathcal{T}_{1,a} = \mathcal{E}_a(v_{a,1})^{t_{1,1}} \mathcal{E}_a(v_{a,2})^{t_{1,2}} \mathcal{E}_a(v_{a,3})^{t_{1,3}} \dots \mathcal{E}_a(v_{a,m})^{t_{1,d}}$ but only one of these components is non-zero because

remember that only one component amongst $v_{a,k}$ is one. Let us suppose, $v_{a,3} = 1$, which means $\mathcal{E}_a(v_{a,3})^{t_{1,3}}$ is non-zero from the first row of T. Therefore, the homomorphic sum of the homomorphic products for the first row will produce $\mathcal{T}_{1,a} = \mathcal{E}_a(v_{a,3})^{t_{1,3}}$, which when decrypted will result in $t_{1,3}$. However, decryption is not done at this stage. If we repeat this for every row in T (with V_a assuming that $v_{a,3} = 1$) and obtain the homomorphic sums per row, we generate a column vector as follows:

$$\mathcal{T}_a = \begin{pmatrix} \mathcal{T}_{1,a} = \mathcal{E}_a(v_{a,3})^{t_{1,3}} \\ \mathcal{T}_{2,a} = \mathcal{E}_a(v_{a,3})^{t_{2,3}} \\ \mathcal{T}_{3,a} = \mathcal{E}_a(v_{a,3})^{t_{3,3}} \\ \cdots \\ \mathcal{T}_{d,a} = \mathcal{E}_a(v_{a,3})^{t_{d,3}} \end{pmatrix} \tag{2}$$

If we homomorphically multiply each element in $\mathcal{T}_a$ with V_b and homomorphically add the resulting components, we get: $\mathcal{T}_b = \mathcal{E}_b(v_{b,1})^{\mathcal{T}_{1,a}} \mathcal{E}_b(v_{b,2})^{\mathcal{T}_{2,a}} \mathcal{E}_b(v_{b,3})^{\mathcal{T}_{3,a}} \ldots \mathcal{E}_b(v_{b,d})^{\mathcal{T}_{d,a}}$ but again, only one of $v_{b,k}$ is non-zero. Let us assume that $v_{b,2} = 1$. Therefore, $\mathcal{T}_b = \mathcal{E}_b(v_{b,2})^{\mathcal{T}_{2,a}}$ because all the other components are effectively zero in plaintext domain. Thus, our 2-D square matrix has been reduced to a point in the encrypted domain, i.e., $\mathcal{T}_b = \mathcal{E}_b(v_{b,2})^{\mathcal{T}_{2,a}}$. If we now run the decryption $\mathcal{D}_b(\mathcal{T}_b)$ first, we effectively obtain $\mathcal{T}_{2,a}$ since $v_{b,2} = 1$. Running the decryption $\mathcal{D}_a(\mathcal{T}_{2,a})$, we obtain $t_{2,3}$, which is exactly the point that can be located by setting $v_{b,2} = 1$ and $v_{a,3} = 1$. Note that for simplicity, we did not describe the shuffling of the components of both vectors because it is related to ensuring randomisation and not the hypercube reduction process. The above process of hypercube reduction can be easily generalised to dimensions higher than $\lambda = 2$. If the encryption function for reducing dimension i to $i-1$ is denoted by $\mathcal{E}_i$ then the ciphertext space for $\mathcal{E}_i$ must be less than the plaintext space of $\mathcal{E}_{i-1}$. In the above example, $\mathcal{E}_i = \mathcal{E}_a$ and $\mathcal{E}_{i-1} = \mathcal{E}_b$. This constraint on the cryptosystems illustrates the fact that with higher dimensions, one would require multiple cryptosystems with large key sizes.

4.1 Computational and Communication Complexities

Assuming that the computational complexity of the combination of a homomorphic addition and a homomorphic multiplication with a cryptosystem that reduces the dimension of the hypercube from i to $i-1$ is c_i. We noticed above that to obtain the final result, we had a complexity of $c_1 d$. Similarly, the complexity due to the hypercube reduction before that was $c_2 d^2$. The total complexity can be expressed as the series $c_1 d + c_2 d^2 + c_3 d^3 + \ldots + c_i d^i + \ldots c_\lambda d^\lambda$. Given n as the total number of elements, we know $n = d^\lambda$. Thus, the expression for complexity can be re-written as $c_\lambda n + c_{\lambda-1} n^{\frac{\lambda-1}{\lambda}} + c_{\lambda-2} n^{\frac{\lambda-2}{\lambda}} + \ldots + c_1 n^{\frac{1}{\lambda}}$, or $\mathcal{O}(n)$. Note that the complexity due to any c_i is higher than that due to any c_j for $i < j$.

The requester sends λ vectors, each of size d while the database responds with a single encrypted value. The sizes of the vectors are different because each contains d values encrypted with different cryptosystems. If we denote the size of a ciphertext for a cryptosystem used to reduce the hypercube from i to

$i-1$ dimension as b_i (independent of n, hence constant) then the total size of the request is $db_1 + db_2 + \ldots + db_\lambda$. The size of the response is always b_1. The communication complexity is in order of λd, or $\log_d(n)\sqrt[\lambda]{n}$. Thus, for an optimal size of d, the complexity is dependent on λ, which means it is in $\mathcal{O}(n)$.

4.2 Reducing the Number of Homomorphic Computations – Impact on Privacy

We noted that the cryptosystem with encryption function $\mathcal{E}_i$ (responsible for reducing the dimension of the hypercube from i to $i-1$) should generate ciphertexts that fit in the plaintext space of the cryptosystem with encryption function $\mathcal{E}_{i-1}$. If $\mathcal{E}_i$ is a 1024-bit Paillier then its ciphertexts are 2048-bits. Therefore, $\mathcal{E}_{i-1}$ must be 2048-bits or above (assuming there is no speciality of the implementations of those cryptosystems, e.g., supporting negative or fractional numbers using plaintext space division). If the 1024-bit Paillier is deemed to be the minimum standard for security then with just $\lambda = 4$, the cryptosystem for E_1 will be the 8192-bit Paillier, which is significantly slow compared to the 1024-bit Paillier. Thus, with higher values of λ, the reductions to certain lower dimensions of the hypercube may not be computationally feasible given the implementations of the cryptosystems.

One way of addressing this challenge is to limit the use of cryptography to only $\lambda = 4$ or $\lambda = 3$, while for all other higher values of λ, the CPIR protocol uses plaintext coordinates to address those dimensions. There is an obvious loss of privacy. According to the original definition of CPIR, the database owner must not know which element (out of n elements) was picked by the requester, thus allowing the requester n-anonymity. If we use the cryptography for reducing only the lower m dimensions, for example, then for any $n = d^\lambda$, the requester will have no privacy in the $d^{\lambda-m}$ dimensions. Suppose $k = d^m$. Then, the requester will still have k-anonymity so long as the plaintext coordinates (for $\lambda - m$ dimensions) are chosen from a uniform random distribution. In other words, as an example, if $m = 4$, and $\lambda = 7$, the requester will specify 3 coordinates in plaintext and use the proposed CPIR protocol for the lower 4 dimensions. Thus, the chosen element will lie somewhere amongst the points in the tesseract defined by d^4. If we assume $d = 100$, the requester will have $k = 100^4$-anonymity.

Loss of Privacy. Following the strategy for quantifying the loss of privacy in [12], we use Shannon's entropy to measure how much privacy is lost for using homomorphic encryption in the m lower dimensions only. The entropy is a measure of uncertainty in a random variable X, and is defined as $\mathcal{H}(X) = -\sum_x p_X(x) \log p_X(x)$. Let $\mathcal{H}(V)$ denote the uncertainty of the vector V, where only one element is 1 and the rest are 0. Since the elements in the vector can either be 0 or 1, and the entire vector can only have one element that is 1, we can write that for a d-element vector, $\mathcal{H}(V) = -\sum_i^d \frac{1}{d} \log \frac{1}{d} = \log d$. Thus, for λ such independent vectors, the total entropy is $\lambda \log d$. If only m such vectors are encrypted, then we can quantify the loss in privacy in terms of entropy as $\mathcal{P}_{loss} = (\lambda - m) \log d$ and leaving us with the residual privacy as $\mathcal{P}_{residual} = m \log d$.

5 Evaluation

In the performance evaluation of cryptographic primitives shown in Table 1, we have used an open-source implementation of the Paillier cryptosystem[1]. The performance of this implementation on a 64-bit Macbook Pro running macOS Sierra 10.12.5 and Java 1.8.0_121-b13 on a 2.9 GHz Intel Core i5 with a 16 GB 2133 MHz LPDDR3 RAM are given in Table 1. The plaintext and integer multiplicands chosen from random integers of bit lengths 256, 512 and 1024 respectively. Notice that these bit lengths are half the size of the public key sizes of the tested cryptosystems because our implementation supports negative integers by dividing the plaintext space into half with the upper half reserved for positive integers and the lower half for negative ones.

Table 1. Comparison of the performances, in terms of time, of a Java implementation of the Paillier cryptosystem with different bit lengths for the public key (i.e. modulus n).

Paillier cryptosystem (key size)	512-bits	1024-bits	2048-bits
Encryption (ms)	1.606	10.586	71.93
Decryption (ms)	1.605	10.773	70.848
Homomorphic addition (ms)	0.156	0.376	1.314
Homomorphic multiplication (ms)	0.894	5.561	36.588

5.1 Computationally Private Information Retrieval (CPIR)

We evaluate the performance of our proposed PIR scheme by setting, without loss of generality, $\lambda = 2$ and each side of our 2-D hypercube to $d = 7$, for simplicity. This involves $d^2 = 49$ homomorphic additions and homomorphic multiplications in $\mathcal{E}_2()$ and only $d = 7$ homomorphic additions and homomorphic multiplications in $\mathcal{E}_1()$. Note that $\mathcal{E}_2()$ is faster than $\mathcal{E}_1()$ because its key size, i.e., n is lower. Setting $\mathcal{E}_2()$ to 512-bits Paillier and $\mathcal{E}_1()$ to 2048-bits Paillier, and using the timings from Table 1 we can compute the time taken for d^2 homomorphic additions and multiplications as $(0.156 + 0.894) \times 49 = 51.45$ ms and for d homomorphic additions and multiplications as $(1.314 + 36.588) \times 7 = 265.314$ ms.

Generalising this, if we denote the time taken by a homomorphic multiplication and a homomorphic addition by $t_{\mathcal{EM}_i}$ and $t_{\mathcal{EA}_i}$ respectively, where $\mathcal{EM}_i$ and $\mathcal{EA}_i$ are applied for hypercube dimension reduction from i to $i-1$, then for a λ-dimensional hypercube represented by $\mathbb{R}$ with each side measuring d, the total time taken to extract a single point due to the homomorphic multiplications and additions is given as:

$$t_{total} = (t_{\mathcal{EM}_\lambda} + t_{\mathcal{EA}_\lambda})d^\lambda + (t_{\mathcal{EM}_{\lambda-1}} + t_{\mathcal{EA}_{\lambda-1}})d^{\lambda-1} + \quad \ldots + (t_{\mathcal{EM}_1} + t_{\mathcal{EA}_1})d$$

Note that every $t_{\mathcal{EM}_i} < t_{\mathcal{EM}_j}$ and every $t_{\mathcal{EA}_i} < t_{\mathcal{EA}_j}$ for $i > j$. Given the optimisation on the CPIR protocol described before, which only uses homomorphic encryption for a limited number of dimension reductions, the time taken will be less than this generalised expression.

[1] Paillier implementation: https://github.com/anirbanbasu/paillier-crypto.

6 Conclusions

In this paper, we have proposed a computationally private information retrieval method based on the concept of hypercubes. We have also shown that in order to make our CPIR scheme efficient, we may need to make limited use of homomorphic cryptosystems with a quantifiable loss of privacy.

One avenue of future work includes testing the proposed system (both versions – one without privacy loss and one with measurable privacy loss) for scalability for picking one token from a large number of tokens in a database.

References

1. Chang, Y.-C.: Single database private information retrieval with logarithmic communication. In: Wang, H., Pieprzyk, J., Varadharajan, V. (eds.) ACISP 2004. LNCS, vol. 3108, pp. 50–61. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-27800-9_5
2. Rivest, R.L., Adleman, L., Dertouzos, M.L.: On data banks and privacy homomorphisms. Found. Secur. Comput. **4**(11), 171–181 (1978)
3. Blakley, G., Meadows, C.: A database encryption scheme which allows the computation of statistics using encrypted data. In: 1985 IEEE Symposium on Security and Privacy, p. 116. IEEE (1985)
4. Abadi, M., Feigenbaum, J., Kilian, J.: On hiding information from an oracle. J. Comput. Syst. Sci. **39**(1), 21–50 (1989)
5. Beaver, D., Feigenbaum, J., Kilian, J., Rogaway, P.: Locally random reductions: improvements and applications. J. Cryptol. **10**(1), 17–36 (1997)
6. Beaver, D., Feigenbaum, J.: Hiding instances in multioracle queries. In: Choffrut, C., Lengauer, T. (eds.) STACS 1990. LNCS, vol. 415, pp. 37–48. Springer, Heidelberg (1990). https://doi.org/10.1007/3-540-52282-4_30
7. Chor, B., Goldreich, O., Kushilevitz, E., Sudan, M.: Private information retrieval. In: Proceedings of the 36th Annual Symposium on Foundations of Computer Science, pp. 41–50. IEEE (1995)
8. Chor, B., Goldreich, O., Kushilevitz, E., Sudan, M.: Private information retrieval. J. ACM **45**(6), 965–982 (1998)
9. Chor, B., Gilboa, N.: Computationally private information retrieval. In: Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing, pp. 304–313. ACM (1997)
10. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48910-X_16
11. Damgård, I., Jurik, M.: A generalisation, a simplification and some applications of paillier's probabilistic public-key system. In: Kim, K. (ed.) PKC 2001. LNCS, vol. 1992, pp. 119–136. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44586-2_9
12. Coney, L., Hall, J.L., Vora, P.L., Wagner, D.: Towards a privacy measurement criterion for voting systems. In: Proceedings of the 2005 National Conference on Digital Government Research, pp. 287–288. Digital Government Society of North America (2005)