International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 05 Issue: 05 | May - 2021 ISSN: 2582-3930

CLOUD BASED EFFICIENT PRIVATION CONJUNCTIVE QUERIES WITH IND-CKA SESURITY

MRS SUCHETA H
Assistant Professor , Dept Of CSE,St. Martin's Engineering College, Secunderabad, Telangana, India

Abstract
A conjunctive query uses the logical conjunction operator and is a special case of a first-order query. First-order queries can be used to write several conjunctive queries. Processing conjunctive queries on public cloud platforms with both keyword and rang conditions while maintaining privacy is a major challenge. No previous privacy-preserving conjunctive query processing scheme based on searchable symmetric encryption (sse) has been able to meet the three requirements of adaptive protection, effective query processing, and scalable index size. In this paper, we introduce the first privacy-preserving conjunctive query processing scheme that meets all three of the above criteria. An indistinguishable bloom filter (ibf) data structure for indexing is proposed to achieve adaptive protection. To achieve efficient query processing and structural indistinguishability, we recommend indistinguishable binary tree, a highly balanced binary tree data structure (ibtree). We suggest an ibtree space compression algorithm to remove redundant information in ibfs in order to achieve scalable and compact index sizes. A traversal minimization algorithm is suggested to improve search efficiency. We suggest updating algorithms to make our scheme dynamic. Using the ind-cka secure model, we show that our scheme is adaptably secured. This paper's main contribution is the achievement of conjunctive query processing with strict privacy guarantees and practical speed and space efficiency. Our tests revealed that our system is both fast and scalable. In the case of millions of records, processing a query takes just a few milliseconds.

Key Terms— Adaptive Ind-Cka Security, Indistinguishable Bloom Filter, Searchable Symmetric Encryption, ,Cloud Computing, Privacy Preserving Conjunctive Queries.

I. Introduction

Cloud computing is a new model for controlling and distributing resources over the internet that has recently gained traction. It's reshaping the information technology landscape and, in the end, making the long-awaited promise of utility computing a reality. With technology advancing and developing at such a rapid rate, it is important to comprehend all facets of it. The anatomy, description, characteristics, affects, design, and core technologies of the Cloud are all covered in this chapter. It categorises Cloud implementation and service models and provides a detailed overview of Cloud service providers.

We're speeding into a new age in which we collect data and conduct costly computations haphazardly on remote servers—the CLOUD, to use a common idiom. The cloud computing paradigm is depicted in Figure 1 with its features, implementation models, service offerings, and components. Although the cloud has many benefits, such as pay-per-use, cost efficiency, accessibility, and a distributed computing environment, it also poses significant concerns regarding data protection, as data stored in the cloud can be harmed by unauthorised user access.

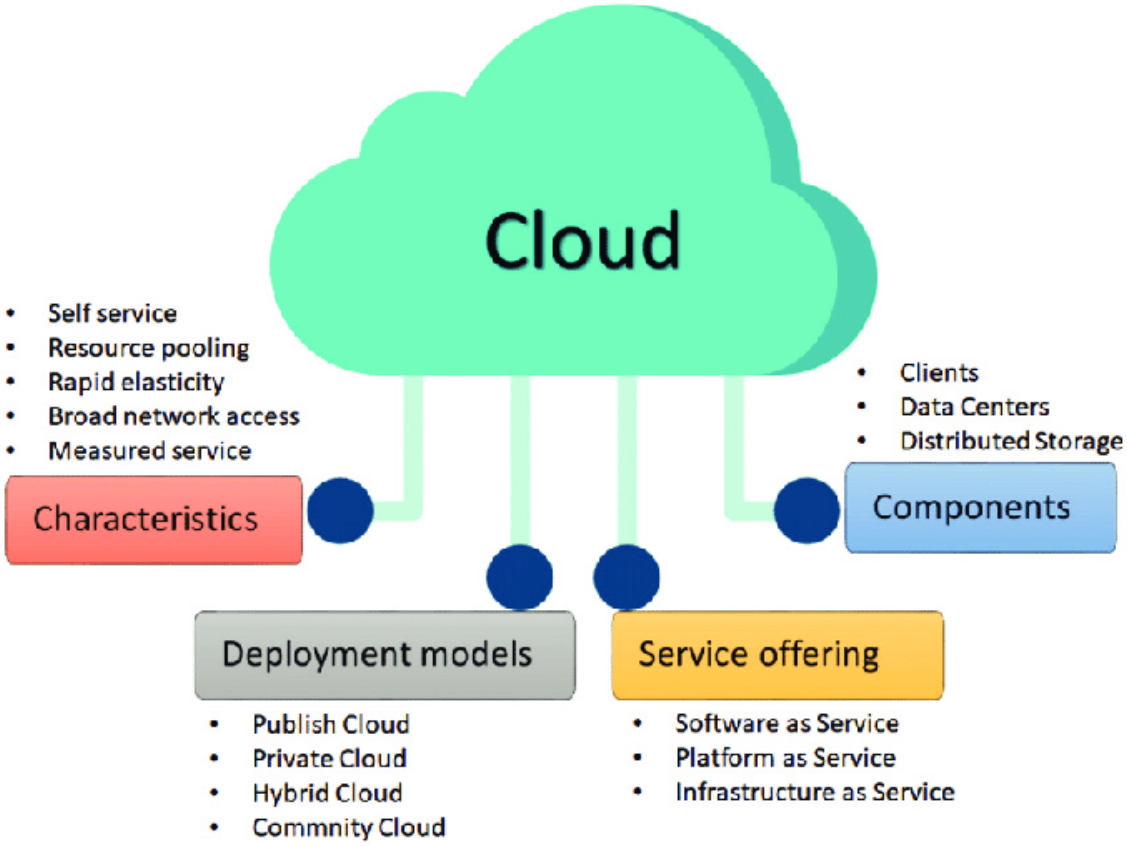

Fig 1: Cloud computing paradigm

SSE (searchable symmetric encryption) allows a client to encrypt data while also allowing it to be searched. SSE's most immediate use is cloud storage, where it allows a customer to safely outsource data to an untrustworthy cloud provider without losing the right to search it. SSE has been the subject of active study, with a variety of schemes proposed to achieve different levels of protection and performance. However, every functional SSE scheme should have the following properties (at a minimum): sublinear search time, protection against adaptive chosenkeyword attacks, compact indexes, and the ability to efficiently add and remove data. Unfortunately, none of the previously identified SSE structures achieves both of these properties simultaneously. SSE's functional utility is extremely limited as a result, and its chances of being deployed in a real-world cloud storage environment are slim.

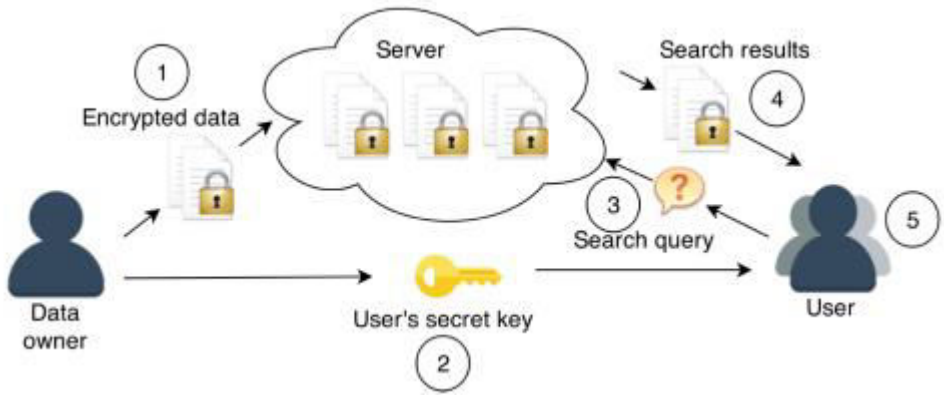

Fig 2 Searchable encryption model.

The basic architecture of a SE scheme is depicted in Figure 2, which consists of four steps: 1. The encryption of Data. The data to be outsourced to the server is encrypted and submitted to the server by the data owner (or a recipient, depending on the scenario). The encrypted data will usually be divided into two parts: (a) Data objects that have been encrypted: Strict key cryptography is usually used to encrypt the data items that the data owner wants to send to the server. A unique identifier is assigned to each encrypted data object (ID). (b) Index that is secure: A secure index is developed, allowing the server to learn the IDs of related encrypted data items and locate them on the server using a search token. 2. Transfer of the user's hidden key. The owner of the data creates a hidden key for a customer and sends it to them. Using their secret key, the user will create search queries. 3. Make a search question. A user (or data owner) creates a keyword search query and sends it to the server, which then decides which encrypted data objects satisfy their search. 4. The outcomes of the quest. The user's search results can be in one of two formats: (a) IDs of encrypted data items that satisfy the search token-associated query; (b) Encrypted data items that satisfy the search token-associated query. In cases where the consumer may not need access to the individual data item (for example, during statistical analysis of encrypted data), the former response is preferable to the latter since it requires lower communication costs. The server locates the encrypted data items satisfying the query using the collection of IDs in the above answer, which increases the search time. In this paper, we look at search results that are made up of IDs of encrypted data objects that only satisfy the query associated with the search token.

II. Related work

The adaptive IND-CKA security model proposed in [8] is used in this paper, and it is by far the best security model for SSE based keyword query schemes. The term "IND-CKA" refers to the index's indistinguishability under chosen keyword attacks, and "adaptive" refers to the adversary's query selection method. The adversary in the adaptive IND-CKA model adaptively selects queries based on previously selected queries, trapdoors, and query outcomes. In the non-adaptive IND-CKA model, on the other hand, the adversary selects all queries at once and obtains the corresponding trapdoors and query results. These two models have different levels of protection. Under the non-adaptive IND-CKA model, a scheme is safe only if the queries are not based on the secure index or previous search results. In contrast, a stable scheme based on the adaptive IND-CKA model provides security even though queries aren't independent of the secure index or previous search results [10].

Adaptive Chosen Keyword Attack Semantic Protection (ind-cka). Intuitively, our security model attempts to catch the idea that an adversary A cannot deduce the contents of a document from its index other than what it already knows from previous query results or other sources. To give you an idea of how the game works, it goes like this: Assume that the challenger C hands the adversary A two equal-length documents V0 and V1, each with a different (possibly unequal) number of words and an index. A's task is to figure out which text is encoded in the index. If distinguishing between the indexes for V0 and V1 is difficult, then deducing at least one of the terms that V0 and V1 do not share from the index must be difficult as well. If A can't say which text is encoded in the index with a probability greater than 1/2, the index doesn't disclose much about its contents. We prove the semantic security of indexes using this formulation of index indistinguishability (ind). It's worth noting that safe indexes don't mask information like document size, which can be gleaned by decrypting the documents. Let's call this index scheme (Keygen, Trapdoor, BuildIndex, SearchIndex). To define semantic defence against an adaptive chosen keyword attack (ind-cka), we use the following game between a challenger C and an attacker A — Prepare for the case. The challenger C generates a set S of q terms for the adversary A to use. A selects multiple subsets from S. 1 S is the list of subsets that is returned to C. C runs Keygen to create the master key Kpriv after obtaining S, and C uses BuildIndex to encode the contents of each subset in S. Finally, C sends all of the indexes to A, along with their corresponding subsets. Inquiries A is permitted to ask C a question about the word x and obtain the trapdoor Tx for x. Tx allows A to use SearchIndex on an index I to see if x I exists.

After performing some Trapdoor queries, A chooses a challenge by selecting a nonempty subset V0 S and generating another non-empty subset V1 from S such that |V0 V1| 6= 0, |V1 V0| 6= 0, and the total length of words in V0 equals that in V1. 2 Finally, A must not have asked C in V04V1 about the trapdoor of any term. Following that, A hands over V0 and V1 to C, who selects b R 0 and 1, calls BuildIndex(Vb, Kpriv), and returns IVb to A. The task for A is to figure out what b is. A is not permitted to ask C for the trapdoors of any word x V04V1 after the challenge has been given. A finally outputs a bit b 0, which represents its best guess for b. AdvA = |Pr[b = b 0] 1/2|, where the likelihood is over A and C's coin tosses.

The adversary is a model of a computational environment's adversarial behaviour. The adversary is usually permitted to influence (or corrupt) some of the parties in a network of communicating parties. The adversary is adaptive if the adversary is allowed to corrupt parties during the computation; otherwise, the adversary is non-adaptive [14]. Since attackers in a computer network can break into computers during the computation, the adaptive adversary models realistic security threats better than the non-adaptive adversary, as pointed out in [14]. If an attacker violates the authentication mechanism during the computation, it may adaptively select queries and execute them to obtain the corresponding results in our computational paradigm.

III. PROPOSAL WORK

The optimized IBtree is IND-CKA Theorem (Security).

L-secure against an adaptive adversary.

Proof: First, we create a simulator S that can generate a simulated index using the information revealed by the leak-age function LS, as shown below. It uses the simulator SE, which is guaranteed to exist by the CPA-security of Enc, to simulate the encrypted data items D = d 1, d2, • • •, dn, as well as the value n and the size of each encrypted data object, which are both included in the performance of LS. It generates an IBtree T according to IBtree Description 4.2 using the identifiers ID = id1, id2, • • •, idn used in the leakage feature to simulate IBtree index I. It then selects m random e-bit strings (s 1, s2, • • •, s m) as m keys for the m twin numbers that will be used to select cells. It's worth remembering that the leakage feature LS includes m. The simulator now creates an IBF Bv with the same size as the corresponding IBF in I for each node v in T. By flipping a coin in the i-th twin of Bv, the simulator stores either 0 at Bv[i][0] and 1 at Bv[i][1], or vice versa. The coin values are also stored locally by the simulator. Finally, the adversary receives this simulated index T from the simulator.

We transform both the deletion and insertion operations into the activity of alteration when upgrading an IBTree. The IBFs in the path from the root to the leaf node that the data item is associated with are reconstructed to alter a data item in an IBTree. As a result, the simulator uses the same method of constructing IBFs mentioned above to generate new IBFs to change the corresponding part of the index T to simulate updating given L U.

The simulator is then used to simulate queries on T. Let Bp stand for the IBF in T's root. If B p[li]=1 and any IBF B in the direction between the root and the leaf,

We say d can be reached by li since B has a position liB that corresponds to li by a random hash function and B[l iB]=1. The data set for position li, denoted as D(li), is the list of all data items that li can access. Let's say the simulator gets a question qi. Due to the revealed search pattern in LQ, the simulator knows if this query has been performed before or not, according to the leakage feature LQ. If it has been done before, the simulator gives the adversary the same trapdoor tqi. If not, the simulator produces a new trapdoor tqi as shown below. Note that a set of location-hash pairs is a query's trapdoor. From the leakage function LQ, the simulator knows the number of location-hash pairs and the query result set Rqi for qi. Assume that the simulator's trapdoors have a limit of p location-hash pairs. The simulator selects p locations and ensures that they meet the conditions D(l1)D(l2)• • •D (lp) = Rqi. The simulator then produces a random string as the hash for each selected location and issues these p location-hash pairs to q I as the trapdoor. Notice that if position l was chosen for previous queries' trapdoors, and each data item d 2 Rqi can be reached by l, then l can also be chosen for qi's trapdoor.

The random oracle can be used to associate a data set D(li) with an unused position li in Bp. The simulator programmes the bit b that the random oracle outputs for a twin in the IBF of a tree node v in the following way: if a data item dx needs to be included in D(li) and v is on the path from the root to the leaf node for dx,

then the simulator lets b be the bit so that Bv[liBv][b]=1 and randomly selects an unused position in both its left Otherwise, if v is not in the path from the root to any leaf node of data item d in D(li) where d 6=dx and the parent node of v is in that path, then the simulator lets b be the bit and chooses no positions in the IBFs of its left and right child nodes to correspond to liBv.

If a probabilistic polynomial time adversary issues a query, the simulator will create a trapdoor for it in the same way as before. Because of the pseudo-random function and CPA-secure encryption algorithm, the adversary cannot distinguish between the trapdoor provided by the simulator and the query result generated by the simulated index T. As a result, our scheme is IND-CKA stable.

IV. RESULT ANALYSIS

Time to Process a Query

The query processing time of IBtree CQ is calculated in milliseconds, according to experimental results. The average query processing time of IBtree CQ increases from 2.2 to 7.4 milliseconds and from 4.7 to 38.1 milliseconds, respectively, when the query result size is set to 50 and the 2 and 3 dimensional data set sizes are increased from 0.5 million to 5 million. When the data set size is set to 5 million, the average query processing time of IBtree CQ increases from 4.1 to 11.1 milliseconds for 2-dimensional data and from 9.1 to 83.2 milliseconds for 3-dimensional data.

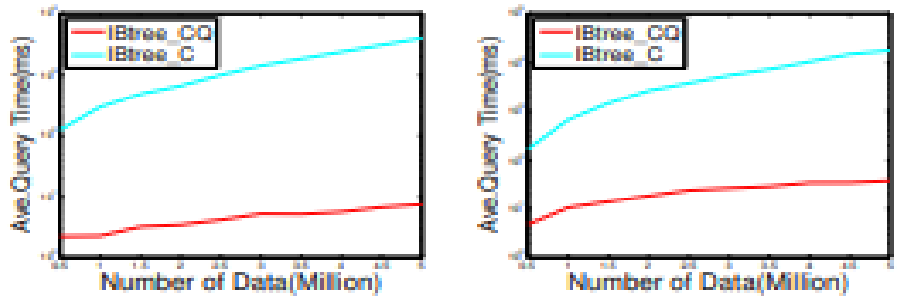

Fig. : Ave. Query Time when |R| = 50

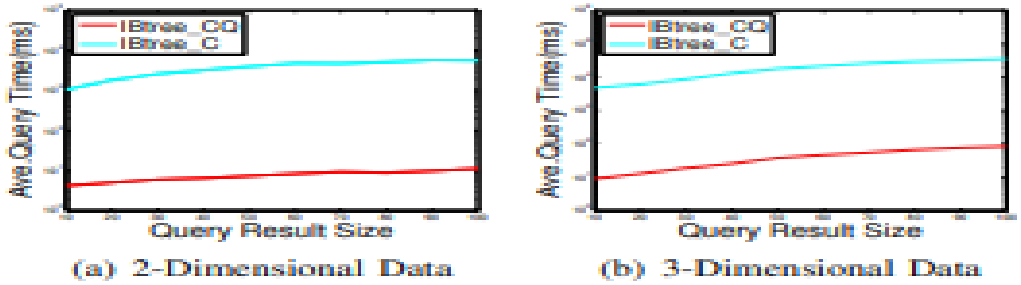

Fig. : Ave. Query Time when $|D|$ = 5 million

Our traversal width and depth minimization algorithms significantly boost search performance, according to experimental results. IBtree CQ consumes orders of magnitude less time when handling queries than IBtree C. When we set the query result size to 50 and increase the size of the data sets from 0.5 million to 5 million, the average query processing time for 2-dimensional data increases from 2.2 to 7.4 milliseconds, while IBtree C increases from 123.7 to 3996.4 milliseconds; for 3-dimensional data, the average query processing time increases from 4.7 to 38.1 milliseconds, whereas IBtree C increases from 166 to 17911 milliseconds. When we set the data set size to 5 million and the result set sizes to 0 to 100, IBtree CQ grows from 4.1 to 11.1 milliseconds for 2-dimensional data, whereas IBtree C grows from 1049.4 to 5621.5 milliseconds for 3-dimensional data; IBtree CQ grows from 9.1 to 83.2 milliseconds for 3-dimensional data, whereas IBtree C grows from 4791 to 34350 milliseconds. Figures and display the average query processing time for these two systems, both using logarithmic coordinates.

CONCLUSIONS

There are four major contributions that we make. To begin, we present the first privacy-preserving conjunctive query processing scheme that meets all three requirements of adaptive protection, efficient query processing, and scalable index size. Several novel concepts, such as IBFs and IBtrees, are incorporated into our system. Second, we present the first probabilistic trapdoor computation algorithm for generating multiple trapdoors for a single query. To achieve space efficiency and query time efficiency, we propose the IBtree space compression algorithm and the IBtree traversal minimization algorithm. Finally, we tested our approach on two real-world data sets. Our scheme is fast in terms of query processing time and scalable in terms of index size, according to experimental results.

REFERENCES

[1] R. Canetti, U. Feige, O. Goldreich, M. Naor. Adaptively secure multi-party computation. STOC, pages 639–648. ACM, 1996.

[2] J. Li and E. R. Omiecinski. Efficiency and security trade-off in supporting range queries on encrypted databases. In Proc. 19th IFIP Conf. on Data and App. Security & Privacy, pages 69–83, 2005.

[3] A. Boldyreva, N. Chenette, and A. O'Neill. Order preserving encryption revisited: Improved security analysis and alternative solutions. In Proc. CRYPTO, pages 578–595, 2011.

[4] D. X. Song, D. Wagner, and A. Perrig. Practical techniques for searches on encrypted data. In Proc. IEEE Symposium on Security and Privacy (S&P), pages 44–55. IEEE, 2000.

[5] E. Goh. Secure indexes. Stanford University Tech. Report, 2004.

[6] D. Boneh, G. D. Crescenzo, R. Ostrovsky, and G. Persiano. Public key encryption with keyword search. In Proc. EUROCRYPT, pages 506–522, 2004.

[7] Y.-C. Chang and M. Mitzenmacher. Privacy preserving keyword searches on remote encrypted data. In Third Inte. Conf. on Applied Cryptography and Network Security (ACNS), pages 442–455, 2005.

[8] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky. Searchable symmetric encryption: improved definitions and efficient con-structions. In Proc. CCS, pages 79-88, 2006.

[9] M. Bellare, A. Boldyreva, and A. O´ıNeill. Deterministic and efficiently searchable encryption. CRYPTO, pages 535–552, 2007.

[10] S. Kamara, C. Papamanthou, and T. Roeder. Dynamic searchable symmetric encryption. In Proc. CCS, pages 965–976, 2012.

[11] D. Cash, S. Jarecki, C. Jutla, H. Krawczyk, M.-C. Rosu, and M. Steiner. Highly-scalable searchable symmetric encryption with support for boolean queries. In CRYPTO, pages 353–373, 2013.

[12] D. Cash, J. Jaeger, S. Jarecki, C. Jutla, H. Krawczyk, M.-C. Rosu, and M. Stiner. Dynamic searchable encryption in very-large databases: Data structures and implementation. In NDSS. 2014.

[13] R. Li, A. X.Liu, L. Wang, and B. Bezawada. Fast range query processing with strong privacy protection for cloud computing. In Proc. VLDB, pages 1953–1964. IEEE, 2014.

[14] R. Canetti , I. Damgard, S. Dziembowski, et al. Adaptive versus Non-Adaptive Security of Multi-Party Protocols. Journal of Cryp-tology 17, pages 153–207, 2004.

[15] V. Pappas, F. Krell, B. Vo, V. Kolesnikov, T. Malkin, S. G. Choi, W. George, A. Keromytis, and S. Bellovi. Blind seer: A scalable private dbms. In Proc. IEEE Symposium on Security and Privacy (S&P), pages 359–374. IEEE, 2014.

[16] B. V. Ben Fisch, F. Krell, A. Kumarasubramanian, V. Kolesnikov, T. Malkin, and S. M.Bellovin. Malicious client security in blind seer: A scalable private dbms. In Proc. IEEE Symposium on Security and Privacy (S&P), pages 395–410. IEEE, 2015.

[17] A. Yao. How to generate and exchange secrets. In Proc. FOCS, pages 162–167. IEEE, 1986.

[18] I. Demertzis, S. Papadopoulos, and O. Papapetrou. Practical private range search revisited. In SIGMOD. pages 185-198, 2016.

[19] P. Karras, A. Nikitin, M. Saad, R. Bhatt, D. Antyukhov, and S. Idreos. Adaptive indexing over encrypted numeric data. In Proc. SIGMOD, pages 171–183. ACM, 2016.

[20] S. Kamara and C. Papamanthou. Parallel and dynamic searchable symmetric encryption. In Proc. Financial Cryptography and Data Security (FC), pages 258–274, 2013.

[21] M. Chase and S. Kamara. Structured encryption and controlled disclosure. In Proc. Theroy and Application of Cryptology and Infor-mation Security(ASIACRYPT), pages 577–594, 2010.

[22] K. Kurosawa and Y. Ohtaki. Uc-secure searchable symmetric encryption. In Proc. Financial Cryptography, pages 285–298, 2012.

[23] P. V. Liesdonk, S. Sedghi, J. Doumen, P. Hartel, and W. Jonker. Computationally efficient searchable symmetric encryption. In Proc. SDM, pages 87–100, 2010.

[24] R. A. Popa, C. M.S.Redfied, N. Zeldovich, and H. Balakrish. Cryptdb: Protecting confidentiality with encrypred query pro-cessing. In Proc. ACM Symposium on Operating Systems Principles (SOSP), pages 85–100. ACM, 2011.

[25] M. Naveed, S. Kamara, and C. V.Wright. Inference attack on property-preserving encrypted databases. In Proc. CCS, pages 644– 655. ACM, 2015.