Characterizing Memory Side Channels in FHE Applications

Abstract—**Privacy-preserving cloud computations ensure accurate operations on encrypted data without revealing sensitive information. Fully Homomorphic Encryption (FHE) allows such computations in the ciphertext space. The result remains in encrypted form and can only be decrypted by using the secret key. Due to its high degree of security, FHE has been adopted widely for applications based on medical or genomic data processing. Though FHE has been shown to be secure, side channels emanating from its operations on underlying hardware have not been thoroughly explored. In this work, we analyze the memory access patterns emanating from FHE-encrypted applications. For an example database query-based application, we show that memory access patterns from different queries can be distinguished by a well-resourced adversary. With the aid of ML-based classifiers, we can predict whether or not a query belongs to an entry in the database, with ∼90% accuracy.**

I. Introduction

Recent advances in cloud computing have facilitated a paradigm shift in data storage on third party servers. Though cloud storage is convenient and cost-effective, sensitive data, such as medical or financial records, can be compromised in such environments. As such, encryption techniques are employed to maintain confidentiality while still allowing efficient retrieval of that data by users. For example, while performing a query involving retrieval of medical records, any leakage of that information is a serious threat to user privacy. To allow query processing on such encrypted data, several techniques have been proposed in searchable symmetric and structural encryption [27], oblivious RAM (ORAM) [13], [14], [28] or fully homomorphic encryption (FHE) [11], [29]. These techniques provide different levels of security based on their threat model of leakage, briefly discussed in [21]. Attacks based on these schemes exploit the data distribution, encryption protocol or even the communication overhead such as in ORAM and FHE [17].

Fully Homomorphic Encryption (FHE). FHE has been proven to be very effective for allowing generic computation on encrypted data, without need for decryption [10]. Figure 1 demonstrates a scenario of querying a record, where the client needs to compute a function f on some private data x. In order to securely do this, the client sends an encrypted version of x, $(E(x))$, to the server. The server then computes $f(E(x))$ over the encrypted data, and sends the result to the client. The client possesses the secret key used for decrypting the result. This process ensures that x is never compromised even when being sent to the server.

Early implementations were based on a Somewhat Homomorphic Encryption (SHE) scheme, which was capable of

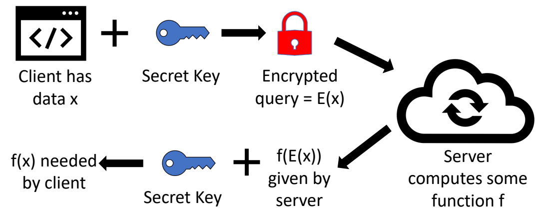


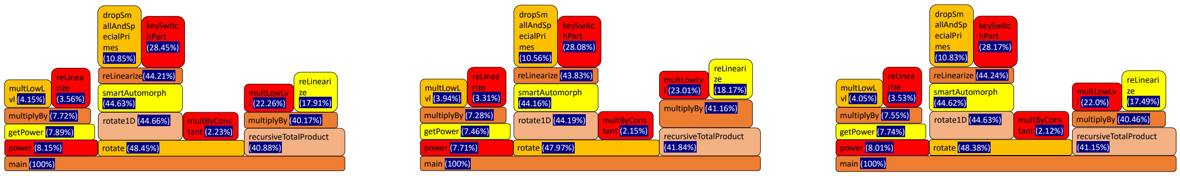
Fig. 1: FHE for outsourced computation

evaluating "low-degree" polynomials involving finite number of operations. A key aspect of HE schemes involve the addition of noise to ciphertext generation. However, this noise accumulates when several operations, especially multiplications and rotations are cascaded one after another, resulting in the data becoming non-recoverable when the noise crosses a certain threshold. To tackle this, Gentry's seminal work on FHE introduced a bootstrapping scheme which can control the noise [10]. BGV [4], BFV [9], GSW [12], TFHE [7], CKKS [5], [6] are some FHE schemes based on lattices and learning-with-errors (LWE) problem, that also control the noise added during encryption. Given the security guarantees provided by the approaches proposed under FHE umbrella, applications based on database queries and machine learning have adapted these to secure their computations [3], [19], [24].

Although FHE has been proven to be secure against mathematical analysis, prior works have shown that it can be vulnerable to system-level attacks such as fault injection [8] and power trace-based side channel attacks [2]. Most database and machine learnng applications operate on huge datasets, which in turn incurs a lot of memory accesses. In this work, we propose analysis of patterns among memory accesses for FHE-encrypted applications to uncover sensitive information. Our threat model is based on the *spatio-temporal distinguishability of memory access patterns*. More specifically, we focus on a *database-query application*, where an adversary analyzes traces generated from two or more unique queries. For a finite set of queries, we show that an attacker can accurately guess whether a query was a valid entry in the database. In addition, we also show that it is possible, with a reasonable degree of confidence, to distinguish between different queries that exist in the database. This attack can be implemented across a range of processor architectures, operating systems and can also extend to different database queries and applications.

II. Threat Model

In this section, we outline the framework used for our preliminary evaluation of memory access patterns as side chan-



(a) Query = Known (Spain) **(b) Query = Known (Estonia)** **(c) Query = Unknown (Mexico)**

Fig. 2: Flamegraph representation of major functions in encrypted search. The percentage denotes that function's contribution to total time

nels to FHE-encrypted applications. FHE implementations are supported by several libraries such as Microsoft SEAL, HElib, Palisade [1], [16], [26]. In this work, we use the Brakerski-Gentry-Vaikuntanathan scheme (BGV) [4] supported by HElib and a database query application adapting this scheme. The search algorithm has several operations on encrypted data as shown in Figure 2, which is a Flamegraph [15] representation of three different queries to the database, two of which (*Spain and Estonia*) exist in the database, while one (*Mexico*) does not. We see that majority of execution time is spent in $power$, $rotate$ and $product$ operations during the search. One interesting observation here is that the amount of time spent in each of the functions does not show much variance across different queries. This implies that simple visual observations would be insufficient in distinguishing between the queries. This motivates us to expand our mode of vulnerability and we delve into memory access traces for gauging leakage.

Our insight is based on the fact that during execution, unique spatial and temporal patterns exist in a memory trace which could render it vulnerable. One such example is timing side channels used by attackers to reveal secret keys used in AES encryption [22]. We collect memory traces from different queries using Pintool [20] and then parse these access traces to identify spatio-temporal regions of interest. To do this, we first determine each spatial memory region and cluster accesses according to their spatial spread. We then determine the temporal distribution of accesses in each spatial region and feed the information into a classifier. The classifier is used to distinguish traces from two sets of queries: 1) those that exist in the database and 2) those that do not. In our experiments, we compare 10 different classifiers as shown in Figure 3. We generate 20 traces for each of 5 unique queries. In this figure, we show that an adversary, namely the classifier, can predict the query class with an accuracy of ∼90% on average. For our second experiment, we collect traces across 12 unique queries. Figure 4 shows that on average, an adversary can distinguish traces from different queries with ∼75% accuracy.

III. Discussion and Future Work

There is much ongoing research in FHE on making it more adaptable for applications and reducing computational overheads [18], [23], [25]. Considering how FHE is becoming more commonplace in real world applications, side channels to the FHE implementation are crucial to privacy. Memory accesses are often used to guide and optimize system design and form an essential part of processor performance verification

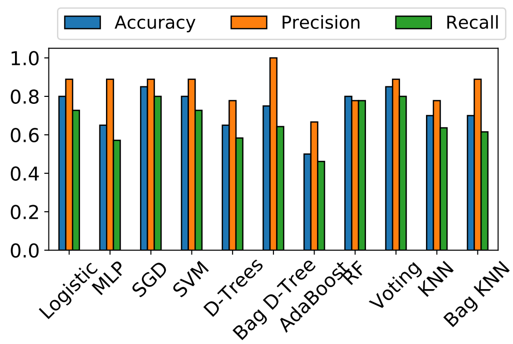


Fig. 3: Accuracy for adversary identifying query for items in the database vs items outside the database

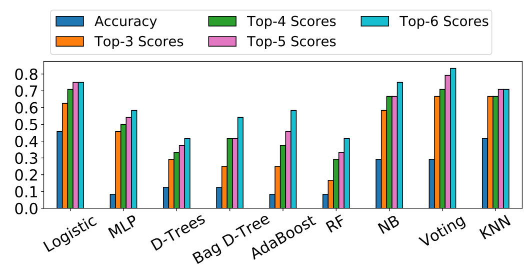


Fig. 4: Accuracy for adversary distinguishing different queries all existing in the database

and monitoring at various stages of design and deployment. Using memory traces from key execution kernels of interest can be used to overcome performance overheads. However, these memory traces poses a serious threat to privacy given how an adversary is able to distinguish between them, even when the data and computations are encrypted using state-of-the-art FHE schemes. In Section II, we demonstrated how memory traces can be exploited as a potential side channel to reveal information. Although we explore a single database query case study for a single FHE scheme (namely BGV), we anticipate that such memory trace-based side channels may be prevalent in other common FHE schemes as well.

Information leakage from memory side channels is a serious problem to which not even post-quantum encryption schemes like FHE are immune. Given FHE primarily focuses on computations happening in third-party servers, effective countermeasures are necessary. As part of future work, we plan to extend our study to privacy-preserving machine learning and other applications, and also focus on developing effective, low-cost mitigative measures.

REFERENCES

[1] "Palisade release," https://gitlab.com/palisade/palisade-release. [Online]. Available: https://gitlab.com/palisade/palisade-release

[2] F. Aydin, E. Karabulut, S. Potluri, E. Alkim, and A. Aysu, "Reveal: Single-trace side-channel leakage of the seal homomorphic encryption library," in *Proceedings of the 2022 Conference and Exhibition on Design, Automation and Test in Europe*, ser. DATE '22. Leuven, BEL: European Design and Automation Association, 2022, p. 1527–1532.

[3] R. Bost, R. A. Popa, S. Tu, and S. Goldwasser, "Machine learning classification over encrypted data," in *22nd Annual Network and Distributed System Security Symposium, NDSS 2015, San Diego, California, USA, February 8-11, 2015*. The Internet Society, 2015. [Online]. Available: https://www.ndss-symposium.org/ndss2015/machine-learning-classification-over-encrypted-data

[4] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(leveled) fully homomorphic encryption without bootstrapping," in *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ser. ITCS '12. New York, NY, USA: Association for Computing Machinery, 2012, p. 309–325. [Online]. Available: https://doi.org/10.1145/2090236.2090262

[5] J. H. Cheon, K. Han, A. Kim, M. Kim, and Y. Song, "Bootstrapping for approximate homomorphic encryption," in *IACR Cryptol. ePrint Arch.*, 2018.

[6] J. H. Cheon, A. Kim, M. Kim, and Y. Song, "Homomorphic encryption for arithmetic of approximate numbers," in *Advances in Cryptology – ASIACRYPT 2017*, T. Takagi and T. Peyrin, Eds. Cham: Springer International Publishing, 2017, pp. 409–437.

[7] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène, "TFHE: Fast fully homomorphic encryption library," August 2016, https://tfhe.github.io/tfhe/.

[8] I. Chillotti, N. Gama, and L. Goubin, "Attacking fhe-based applications by software fault injections," Cryptology ePrint Archive, Paper 2016/1164, 2016, https://eprint.iacr.org/2016/1164. [Online]. Available: https://eprint.iacr.org/2016/1164

[9] J. Fan and F. Vercauteren, "Somewhat practical fully homomorphic encryption," *IACR Cryptol. ePrint Arch.*, vol. 2012, p. 144, 2012.

[10] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*, ser. STOC '09. New York, NY, USA: Association for Computing Machinery, 2009, p. 169–178. [Online]. Available: https://doi.org/10.1145/1536414.1536440

[11] C. Gentry, "Computing arbitrary functions of encrypted data," *Commun. ACM*, vol. 53, no. 3, p. 97–105, mar 2010. [Online]. Available: https://doi.org/10.1145/1666420.1666444

[12] C. Gentry, A. Sahai, and B. Waters, "Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based," in *Advances in Cryptology – CRYPTO 2013*, R. Canetti and J. A. Garay, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 75–92.

[13] O. Goldreich, "Towards a theory of software protection and simulation by oblivious rams," in *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, ser. STOC '87. New York, NY, USA: Association for Computing Machinery, 1987, p. 182–194. [Online]. Available: https://doi.org/10.1145/28395.28416

[14] O. Goldreich and R. Ostrovsky, "Software protection and simulation on oblivious rams," *J. ACM*, vol. 43, no. 3, p. 431–473, may 1996. [Online]. Available: https://doi.org/10.1145/233551.233553

[15] B. Gregg, "The flame graph: This visualization of software execution is a new necessity for performance profiling and debugging." *Queue*, vol. 14, no. 2, p. 91–110, mar 2016. [Online]. Available: https://doi.org/10.1145/2927299.2927301

[16] S. Halevi and V. Shoup, "Design and implementation of helib: a homomorphic encryption library," Cryptology ePrint Archive, Paper 2020/1481, 2020, https://eprint.iacr.org/2020/1481. [Online]. Available: https://eprint.iacr.org/2020/1481

[17] G. Kellaris, G. Kollios, K. Nissim, and A. O'Neill, "Generic attacks on secure outsourced databases," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 1329–1340. [Online]. Available: https://doi.org/10.1145/2976749.2978386

[18] S. Kim, J. Kim, M. J. Kim, W. Jung, J. Kim, M. Rhu, and J. H. Ahn, "Bts: An accelerator for bootstrappable fully homomorphic encryption," ser. ISCA '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 711–725. [Online]. Available: https://doi.org/10.1145/3470496.3527415

[19] T. Li, Z. Huang, P. Li, Z. Liu, and C. Jia, "Outsourced privacy-preserving classification service over encrypted data," *Journal of Network and Computer Applications*, vol. 106, pp. 100–110, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1084804517304356

[20] C.-K. Luk, R. Cohn, R. Muth, H. Patil, A. Klauser, G. Lowney, S. Wallace, V. J. Reddi, and K. Hazelwood, "Pin: Building customized program analysis tools with dynamic instrumentation," in *Proceedings of the 2005 ACM SIGPLAN Conference on Programming Language Design and Implementation*, ser. PLDI '05. New York, NY, USA: ACM, 2005, pp. 190–200. [Online]. Available: http://doi.acm.org/10.1145/1065010.1065034

[21] M. Naveed, "The fallacy of composition of oblivious ram and searchable encryption," *IACR Cryptology ePrint Archive*, vol. 2015, p. 668, 2015. [Online]. Available: https://eprint.iacr.org/2015/668

[22] D. A. Osvik, A. Shamir, and E. Tromer, "Cache attacks and countermeasures: the case of aes," in *Cryptographers' track at the RSA conference*. Springer, 2006, pp. 1–20.

[23] B. Reagen, W.-S. Choi, Y. Ko, V. T. Lee, H.-H. S. Lee, G.-Y. Wei, and D. Brooks, "Cheetah: Optimizing and accelerating homomorphic encryption for private inference," in *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2021, pp. 26–39.

[24] T. K. Saha, M. Rathee, and T. Koshiba, "Efficient private database queries using ring-lwe somewhat homomorphic encryption," *Journal of Information Security and Applications*, vol. 49, p. 102406, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2214212619303345

[25] N. Samardzic, A. Feldmann, A. Krastev, S. Devadas, R. Dreslinski, C. Peikert, and D. Sanchez, "F1: A fast and programmable accelerator for fully homomorphic encryption," in *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 238–252. [Online]. Available: https://doi.org/10.1145/3466752.3480070

[26] "Microsoft SEAL (release 4.0)," https://github.com/Microsoft/SEAL, Mar. 2022, microsoft Research, Redmond, WA.

[27] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proceeding 2000 IEEE Symposium on Security and Privacy. S P 2000*, 2000, pp. 44–55.

[28] E. Stefanov, M. van Dijk, E. Shi, C. Fletcher, L. Ren, X. Yu, and S. Devadas, "Path ORAM: An extremely simple oblivious ram protocol," in *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, ser. CCS '13, 2013, p. 299–310.

[29] V. Vaikuntanathan, "Computing blindfolded: New developments in fully homomorphic encryption," in *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, 2011, pp. 5–16.