Demystifying the comments made on "A Practical Full Key Recovery Attack on TFHE and FHEW by Inducing Decryption Errors"

Bhuvnesh Chaturvedi, Anirban Chakraborty, Ayantika Chatterjee, and Debdeep Mukhopadhyay

Indian Institute of Technology, Kharagpur
bhuvneshchaturvedi2512, ch.anirban00727, cayantika, debdeep.mukhopadhyay@gmail.com

18 December 2022

Abstract. Fully Homomorphic Encryption (FHE) allows computations on encrypted data without the need for decryption. Therefore, in the world of cloud computing, FHE provides an essential means for users to garner different computational services from potentially untrusted servers while keeping sensitive data private. In such a context, the security and privacy guarantees of well-known FHE schemes become paramount. In a research article, we (Chaturvedi et al., ePrint 2022/1563) have shown that popular FHE schemes like TFHE and FHEW are vulnerable to CVO (Ciphertext Verification Oracle) attacks, which belong to the family of "reaction attacks" [6]. We show, *for the first time, that feedback from the client (user) can be craftily used by the server to extract the error (noise) associated with each computed ciphertext.* Once the errors for some m ciphertext ($m > n$, where n = key size) are retrieved, the original secret key can be trivially leaked using the standard Gaussian Elimination method. The results in the paper (Chaturvedi et al., ePrint 2022/1563) show that FHE schemes should be subjected to further security evaluations, specifically in the context of system-wide implementation, such that CVO-based attacks can be eliminated. Quite recently, Michael Walter published a document (ePrint 2022/1722), claiming that the timing channel we used in our work (Chaturvedi et al., ePrint 2022/1563) "are false". In this document, we debunk this claim and explain how we use the timing channel to improve the CVO attack. We explain that the CVO-based attack technique we proposed in the paper (Chaturvedi et al., ePrint 2022/1563) is a result of careful selection of perturbation values and the first work in literature that showed reaction based attacks are possible in the context of present FHE schemes in a realistic cloud setting. We further argue that for an attacker, any additional information that can *aid* a particular attack shall be considered as leakage and must be dealt with due importance to stymie the attack.

Keywords: FHE · TFHE · FHEW · CVO attack.

1 Introduction

Fully Homomorphic Schemes (FHE) are a class of encryption schemes that allow arbitrary computations on encrypted data without the need to decrypt it first, while also ensuring that the output remains encrypted as well. Such schemes are helpful in the construction of privacy-preserving protocols in the cloud computation scenario that allows a user (client) to offload its confidential data onto the remote cloud, which can also perform arbitrary computations on it on behalf of the user without revealing the original data. The basic assumption of FHE is that the server is untrusted, and thus it should not know any information about the client data. This stems from the fact that the data stored on the server is encrypted under the client's key which the server does not possess. On the other hand, the server is free to perform any computations on the encrypted data as it is not under the client's control. Therefore, considering the underlying crypto-primitive to be mathematically strong, an untrusted or malicious server may start undertaking spurious activities including, but not limited to, manipulating client's data in order to extract private information. In short, the aim of such a server is to retrieve private information about the client's data while also ensuring that the attack remains undetected, so as not to lose trust of the client. As such a scenario is totally valid under the assumptions of FHE, it becomes necessary to evaluate the security of well-known FHE schemes from the practical aspect as well, apart from primitive and implementation levels.

In the research article [2], published on 2022-11-18 (`https://ia.cr/2022/1563`), we showed a *practical end-to-end attack* on two FHE schemes, TFHE [3] and FHEW [5], using the assumptions that a malicious server can introduce calculated perturbations in the computed ciphertext and on receiving *feedback* from the client, can effectively retrieve the underlying error terms for each corresponding ciphertext. This led to complete recovery of the secret key for both TFHE and FHEW, in realistic timeframe. As correctly pointed out by Michael Walter in [9], "Considering the typical usecase of FHE, this seems devastating." In the next section, we briefly discuss about the attack procedure and the clever use of "timing channel" for both the libraries.

2 CVO attack on TFHE and FHEW

In this section, we provide a short summary of the attack presented in [2]. For a more detailed understanding, please refer to the original document at `https://ia.cr/2022/1563`.

At first let us clarify the security assumptions we considered in our attack. FHE schemes are assumed to be IND-CPA secure, which means an attacker may obtain encryptions on arbitrarily chosen plaintext. However, they are neither IND-CCA nor IND-CCA2 secure, meaning that availability of decryption oracle will essentially break the FHE schemes. Our attack [2] operates under the notion of IND-CVA (Indistinguishability against Ciphertext Verification Attack) security [4], which is based on the idea of "reaction" attack from [6]. Under this premise, it is assumed that an adversary has access to an oracle that *accepts a ciphertext as input and returns as output whether the decryption was successful*

or not. This oracle, which we refer to as Ciphertext Verification Oracle or CVO, is essentially the client itself in a "pay per running times model", where client pays for each correct computation [4]. In such a model, the client could ask for a free re-computation in case the result returned by the cloud is incorrect. The client before using the FHE cloud services would typically have a verification phase, wherein it will check the correctness of the homomorphic ciphertexts. In case of decryption failures the client would need to report the same to the cloud, to avoid payments for erroneous service of the cloud. Therefore, contrary to the claim made by Michael Walter [9], CVO access is not a CCA2-style attack [1]. Moreover, applicability of IND-CVA approach on FHE schemes has been reported earlier in literature [4, 7, 8, 10].

The author in [9] provides an over-simplified view of the entire attack, which we feel, does not properly highlight or justifies the intricacies of the attack and the challenges faced for executing the attack in a practical scenario. We will provide a detailed explanation of the attack in this document. The premise of the attack is that a ciphertext can be decrypted correctly at the client's end only if the error lies within the error threshold. Therefore, if the server is able to carefully perturb the ciphertext with additional errors, it can "invert" the decrypted bit. While the malicious server could perturb the ciphertext outputs to instigate reaction from the client, there exist two major challenges that the server needs to deal with. ① the knowledge of the plaintext value for the corresponding ciphertext and ② the sign of the actual error. In our paper [2] (section 5.2, page 6), we explain how the server can manipulate the induced perturbations in order to first recover the error sign and then the original plaintext message. One must also note that, the procedure for recovering of the plaintext message is different for TFHE and FHEW. For TFHE we subtract 2μ, where $\mu = 2^{29}$, from the ciphertext, to invert the plaintext message and determine the original value from the reaction of the client. For FHEW, the plaintext recovery process involves additional gate operations on the originally computed ciphertext such that the resultant output always decrypts to a 0. It must be mentioned that this is the first work in literature that provides a complete step-by-step methodology to recover the underlying error values for both TFHE and FHEW. Once the sign and plaintext message are obtained, the exact error value can be recovered using the CVO by craftily manufacturing perturbations and implementing a binary search-based algorithm. A detailed explanation and step-by-step procedure can be found in the original paper. We have also provided links to the publicly available codes to test and verify the attack.

3 Timing Channel in both TFHE and FHEW

In our preliminary report (ePrint 2022/685) [1], we found out that the computation time for homomorphic gate operations in TFHE has a relationship with the error values in the computed ciphertext. Therefore, given a timing range (t_1, t_2),

[1] We have provided a subsection in [2] (section5, page 5) where we discuss ***Why Decryption Verification Oracle is not a Decryption Oracle?***. Interested readers are encouraged to peruse the arguments presented in that section.

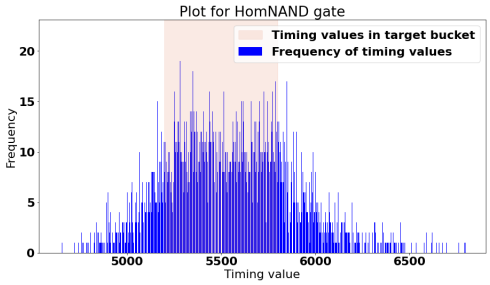

Fig. 1. Plot of timing values of *10k* ciphertexts. The shaded region highlights the timing values that correspond to the timing bucket containing most number of errors.

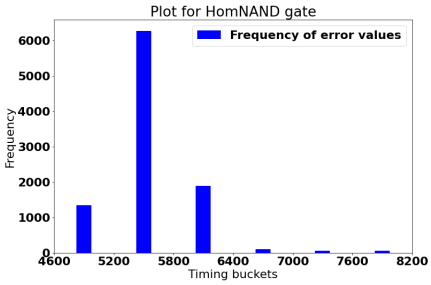

Fig. 2. Plot of count of errors in each timing bucket. The bucket with the highest count is our bucket of interest, which corresponds to timing values between 5200 and 5800.

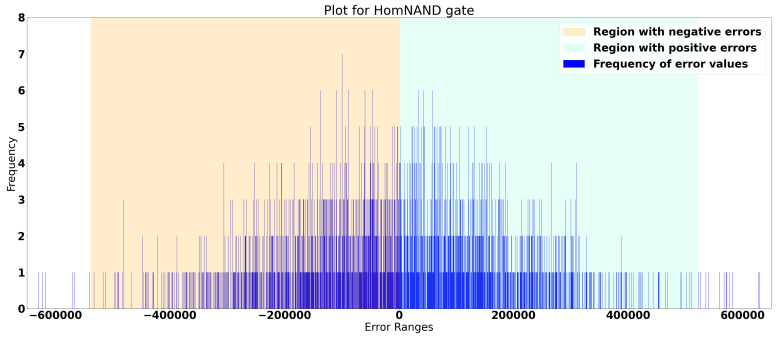

Fig. 3. Plot of frequency of errors in intervals of 100. The shaded regions highlights the bound of errors in the targeted bucket, the left region (highlighted in yellow) is for the negative errors while the right region (highlighted in blue) is for the positive errors.

the range of error values can be predicted with a high probability. We created *templates* of the error values by running the gate operations $100k$ times. Using these templates, we found that the overall range of the possible error values could be reduced to 2^{21}. This was a preliminary result and did not contain a full attack. We published the full attack in [2], assuming a CVO access to the server where we used active perturbations to the computed ciphertext to entice the client to send feedback when the decryption goes wrong. It must be mentioned that this "reaction-based attack" does not require the use of timing information. In the Github repository (`https://github.com/SEAL-IIT-KGP/CVO-TFHE`) available in the paper [2], *we have provided source code where the full end-to-end attack works, without the use of timing channel*. The codes were made public on 11 November 2022 (a month before the report by Walter [9] was published).

We use the timing information (templates or bins or buckets, as one might call them) to effectively filter out the error values that are very less likely to appear. As pointed out in [9], in TFHE, the error is sampled from a "Gaussian-like" distribution with standard deviation $\sigma = 2^{17}$ and centered around 0. Therefore, the error values towards the *tails* of the Gaussian distribution are less likely to

appear when randomly drawn. We essentially use this information to choose the perturbation error values. In our attack [2], we perturb the output ciphertext such that the plaintext message will always be 1 for both TFHE and FHEW and the error +ve for all the perturbed samples. Without the help of timing information, if we consider e_{min} as 0 and e_{max} as 2^{31}, we require 36 queries per ciphertext. However, using the timing information, we are able to reduce that number to 22 queries. It's a significant improvement as *it reduces the total number of queries for the entire key retrieval by almost 10k*. It must be mentioned here that we utilize the timing information to make the attack efficient and converge faster. The timing information is not a necessity but rather an optimization technique, in this case.

Let us understand how and why the timing channel works. The timing distribution in Fig. 1 (in blue) shows the frequency of each timing value obtained while running homomorphic NAND operation $10k$ times on two ciphertexts in TFHE. The highlighted portion shows the range of timing values that fell in the highest bucket (bucket with the most error values). Fig.2 shows the distribution of error values in each timing bucket. One can note that only one bucket contains the maximum number of error values while others contain considerably lesser. Finally, Fig. 3 shows the frequency of errors in the overall sample ($10k$ runs) in blue. The two highlighted section shows the error ranges (-e_{max}, -e_{min}) (in yellow) and (+e_{min}, +e_{max}) (in cyan), as ascertained from the highest bucket. One can clearly observe that the highlighted portions reduce the search space for the error, aided by the timing information. We have explained this phenomenon to answer one of the questions from a reviewer in a respected security conference a couple of months back. The exact answer provided during the rebuttal phase (October 24 - November 7, 2022) is provided below.

We perform templating by generating 10k ciphertext pairs with arbitrarily chosen key (s) and recording their execution time for NAND-operation. We observed that, given Gaussian nature of the timing distribution, the median range (5100 - 5600 cycles for our case) contains >75% errors, while others contain the rest. We choose the least and highest error-values as e_{min} and e_{max}. During the attack phase using an unknown key and random ciphertext, we record the time and observe that most ciphertexts fall into the median bucket. As errors are randomly-sampled from a Gaussian distribution, most of the errors during the attack-phase lie within the range (e_{min}, e_{max}) that we determined during templating. In case an error lies outside the given range, our binary-search algorithm would not be able to ascertain the correct value of the error, as it doesn't converge and eventually, we ignore such ciphertexts. Templates formed are independent of the secret key due to stochastic equivalence wrt. keys. Therefore, the template building can be done using a randomly chosen secret key 's' while the same templates can be used to mount attacks on ciphertexts generated using any random key.

Improvement suggested by Walter [9]

Michael Walter suggested an improvement over the CVO attack where he showed that the number of queries can be reduced to 22 without the timing information. Similar to our claim, the reduction is possible due to the Gaussian nature of the distribution from which the error values are sampled. Although he showed that a bound of 2^{21} can be assumed, it does not provide a concrete value on the minimum and maximum error values that are required for the CVO attack (our attack) to work. In contrast, we use the timing information as a *first level filter* to start with a smaller search space which in turn leads to faster convergence. If any error value, during the attack phase, lies beyond the range (e_{min}, e_{max}), the corresponding ciphertext will be automatically discarded by the server by observing the timing value. This approach helps in filtering out the outlier values early, thereby saving a number of queries. On the other hand, using Walter's approach, one will end up using 21 queries but eventually won't retrieve the error as the error lies beyond the chosen range. We agree that Walter's approach can also lead to successful key retrieval using our CVO-based attack method with 22 queries per ciphertext. However, we note that our approach using the timing information and templating *is the first attempt in known literature* that provides an end-to-end attack on both TFHE and FHEW in a realistic setting and timeframe. As the attacker (server) uses timing information to perform the attack, we feel the attack must be considered as a combination of both "reaction and timing"-based attack. The approach proposed by Walter [9] can be considered as an alternative to the timing-based attack that we presented.

4 Conclusion

In this report, we discuss the attack presented by Chaturvedi et. al. [2] on TFHE and FHEW and their practicality in the context of cloud computing. We establish that our proposed attack in [2] does not require the use of a timing channel for key retrieval for both TFHE and FHEW. However, we use the timing information to reduce the search space of the errors and hasten the key-retrieval process. This in turn leads to much lesser number of overall queries made to the client, thereby lowering the chances of raising suspicion by the user. An attacker does not follow any rules or conventional path. Anything that lies within the realistic threat model can be and will be used by an attacker to unearth the secret information. Since the availability of the timing channel provides an advantage to the attacker, it must be considered a potential threat and any countermeasure that is designed must take this into effect. Therefore, as security researchers, we must consider all the leakage sources while designing important security schemes like FHE. As Walter correctly pointed out *"This attack serves as a stark reminder that for deployment, FHE needs to be securely embedded into a higher level protocol that ensures the security of participants against realistic attackers".* Therefore, the onus is on the FHE designers to provide a secure leakage-free implementation such that these schemes can be used for the larger good.

References

1. Chaturvedi, B., Chakraborty, A., Chatterjee, A., Mukhopadhyay, D.: Error leakage using timing channel in fhe ciphertexts from tfhe library. Cryptology ePrint Archive, Paper 2022/685 (2022), https://eprint.iacr.org/2022/685, https://eprint.iacr.org/2022/685
2. Chaturvedi, B., Chakraborty, A., Chatterjee, A., Mukhopadhyay, D.: A practical full key recovery attack on tfhe and fhew by inducing decryption errors. Cryptology ePrint Archive, Paper 2022/1563 (2022), https://eprint.iacr.org/2022/1563, https://eprint.iacr.org/2022/1563
3. Chillotti, I., Gama, N., Georgieva, M., Izabachène, M.: TFHE: fast fully homomorphic encryption over the torus. J. Cryptol. **33**(1), 34–91 (2020). https://doi.org/10.1007/s00145-019-09319-x, https://doi.org/10.1007/s00145-019-09319-x
4. Chillotti, I., Gama, N., Goubin, L.: Attacking fhe-based applications by software fault injections. Cryptology ePrint Archive, Paper 2016/1164 (2016), https://eprint.iacr.org/2016/1164, https://eprint.iacr.org/2016/1164
5. Ducas, L., Micciancio, D.: Fhew: Bootstrapping homomorphic encryption in less than a second. In: Oswald, E., Fischlin, M. (eds.) Advances in Cryptology – EUROCRYPT 2015. pp. 617–640. Springer Berlin Heidelberg, Berlin, Heidelberg (2015)
6. Hall, C., Goldberg, I., Schneier, B.: Reaction attacks against several public-key cryptosystem. In: Varadharajan, V., Mu, Y. (eds.) Information and Communication Security. pp. 2–12. Springer Berlin Heidelberg, Berlin, Heidelberg (1999)
7. Hu, Z., Sun, F., Jiang, J.: Ciphertext verification security of symmetric encryption schemes. Sci. China Ser. F Inf. Sci. **52**(9), 1617–1631 (2009)
8. Loftus, J., May, A., Smart, N.P., Vercauteren, F.: On cca-secure somewhat homomorphic encryption. In: In Selected Areas in Cryptography. pp. 55–72 (2011)
9. Walter, M.: On side-channel and cvo attacks against tfhe and fhew. Cryptology ePrint Archive, Paper 2022/1722 (2022), https://eprint.iacr.org/2022/1722, https://eprint.iacr.org/2022/1722
10. Zhang, Z., Plantard, T., Susilo, W.: Reaction attack on outsourced computing with fully homomorphic encryption schemes. In: Kim, H. (ed.) Information Security and Cryptology - ICISC 2011. pp. 419–436. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)