On Side-Channel and CVO Attacks against TFHE and FHEW

Michael Walter

Zama, France

December 14, 2022

Abstract

The recent work of Chaturvedi *et al.* (ePrint 2022/685) claims to observe leakage about secret information in a ciphertext of TFHE through a timing side-channel on the (untrusted) server. In (Chaturvedi *et al.*, ePrint 2022/1563) this is combined with an active attack against TFHE and FHEW. The claims in (Chaturvedi *et al.*, ePrint 2022/685) about the non-trivial leakage from a ciphertext would have far-reaching implications, since the server does not have any secret inputs. In particular, this would mean a weakening of LWE in general, since an adversary could always simulate a server on which there is side channel leakage.

In this short note, we show that the claims made in the two aforementioned works with regards to the leakage through the timing side channel are false. We demonstrate that the active attack, a standard attack against IND-CPA secure LWE-based encryption, can be mounted just as efficiently without the "side channel information".

1 Introduction

The paper [CCCM22b] claims an efficient key-recovery attack on TFHE [CGGI20] and FHEW [DM15] i.e., an attack in which the server can efficiently recover the secret key of the client. Considering the typical usecase of FHE, this seems devastating. The attack consists of two parts, which we will discuss in this note.

1) The server is provided a Ciphertext Verification Oracle (CVO), which means it may submit arbitrary ciphertexts to the client and in some way learns if the ciphertext decrypted correctly or not. This oracle can be used to extract the noise in the ciphertexts. Using noise-free ciphertexts it is straight-forward to recover the secret key.

2) Building on work from [CCCM22a], the server may use a timing side channel on itself during the execution of homomorphic operations to learn some non-trivial information about the noise in the ciphertext. This is meant to speed up the attack in 1).

We briefly (and informally) recall some basic security notions: encryption schemes secure against Chosen Plaintext Attacks (CPA) are secure against attackers that may obtain encryptions of arbitrary plaintexts, but do not have access to a decryption oracle; i.e., attackers that only observe ciphertexts of plaintexts of their choice and try to learn some information from them. It is well known that in practice, IND-CPA security is generally not sufficient. Rather, one should deploy encryption schemes that are also secure against attackers, which may tamper with ciphertexts and obtain decryptions of them. Such attacks are called Chosen Ciphertext Attacks (IND-CCA2, where the *2* indicates that the attacker may choose its queries *adaptively*). There are generic transforms from IND-CPA secure schemes to IND-CCA2 secure schemes; e.g., [FO99]. Unfortunately, none of them apply to FHE schemes, since the transformation will destroy the homomorphic properties of the scheme. In fact, it is easy to see that no FHE scheme can possibly be IND-CCA2 secure.[1]

Take-aways It is not hard to see that a CVO oracle yields a CCA2-style attack and is indeed devastating to IND-CPA secure LWE-based encryption. This attack serves as a stark reminder that for deployment, FHE needs to be securely embedded into a higher level protocol that ensures security of participants against realistic attackers.

On the other hand, we show below that there is no evidence that there is any leakage of the noise on the server side and any insinuation that TFHE or FHEW could be weakened by some side channel on the server should be considered unsubstantiated.

[1] Whether efficient FHE schemes can be IND-CCA1 secure (against non-adaptive chosen ciphertext attacks) is unclear. See [FHR21] for an overview of the current state of affairs.

2 CVO Attack on LWE

We first give a description of the basic attack from [CCCM22b].

Basic LWE Encryption Consider a typical LWE ciphertext $(\mathbf{a}, b = \langle \mathbf{a}, \mathbf{s} \rangle + \mu + e)$ with ciphertext modulus $q \in \mathbb{Z}$ and key length $n \in \mathbb{Z}$, where

- $\mathbf{a} \xleftarrow{\$} \mathbb{Z}_q^n$;
- $\mathbf{s} \leftarrow \Psi$ where Ψ is some distribution over $\mathbb{Z}_q^n$ with large enough entropy;
- $e \leftarrow \chi$, where χ is some distribution on $\mathbb{Z}$ concentrated around 0 and with standard deviation σ;
- $\mu \in \mathbb{Z}_q$ is an encoding of a message $m \in \{0, 1\}$ such that m can be recovered from $\mu + e$.

The reader may think of $\mu = \frac{q}{2}m$ and recovering the message as outputting $m = 1$ if $\mu + e \in [q/4, 3q/4]$ and 0 otherwise. Recall that the standard deviation σ of the noise e is crucial for security: if σ was 0, an attacker with knowledge of m could recover the secret key $\mathbf{s}$ from many "encryptions" of m simply by solving a linear system over $\mathbb{Z}_q$. The larger σ the more secure the cryptosystem. For large enough σ, LWE-based encryption is considered to be IND-CPA secure. How large σ needs to be exactly for a certain level of security depends on the other parameters, in particular the secret distribution Ψ and dimension n and the ciphertext modulus q. A useful tool to quickly evaluate the security of LWE parameters is the lattice estimator [APS15].[2] Of course, σ cannot be set arbitrarily large, since it needs to be possible to recover the message despite the noise. In the example setting above, note that the noise needs to be smaller than $q/4$ in absolute value in order for the decoding of $\mu + e$ to be correct. In other words, the standard deviation allows controlling the trade-off between security (in combination with other parameters) and correctness of the scheme.

In the following we will concentrate on parameters in the typical TFHE setting, since the attack mainly focuses on TFHE, which are $q = 2^{32}$, $n = 630$, Ψ is the uniform distribution over binary vectors and χ is a "Gaussian-like" distribution centered around 0 with $\sigma = 2^{17}$. The message encoding is slightly different and changes during homomorphic operations,

[2] https://github.com/malb/lattice-estimator

but for simplicity we consider the above encoding. Everything here easily generalizes to other encodings.

CVO Attack Assume an attacker has a CVO oracle at its disposal, which allows to check for any δ and ciphertext $(\mathbf{a}, b)$ encrypting m if $(\mathbf{a}, b + \delta)$ still encrypts m. It is then straight-forward to recover the noise e in the ciphertext using binary search: check if $(\mathbf{a}, b + q/4)$ still encrypts m. If it does, the noise e was negative, otherwise positive. Depending on the outcome, check if $(\mathbf{a}, b + 5q/8)$ or $(\mathbf{a}, b + q/8)$ still encrypts m to narrow down the noise further, etc. So with $\log_2(q/2)$ queries to the oracle the attacker may obtain the noise of a ciphertext and thus remove it and obtain an noise-free ciphertext. If, for example, $q = 2^{32}$ as in TFHE, then the noise can be recovered using 31 queries. An attacker may repeat this for n many ciphertexts, which allows to recover the key using linear algebra. This is the attack proposed in [CCCM22b].

We can actually improve on this attack. Note that above the specific distribution χ of the noise e is ignored, even though the attacker typically knows it. The attack can easily be adapted to require only $\lceil \log_2(2B) \rceil$ queries per ciphertext if the noise can be bounded as $|e| < B$. For example, for the "Gaussian-like" distribution in TFHE with standard deviation $\sigma = 2^{17}$ and centered around 0, we can put a bound on the size of the noise such that n ciphertexts have noise within this bound with high probability. Making the heuristic assumption that χ approximately behaves like a standard Gaussian distribution with standard deviation 2^{17}, the probability that a single sample from χ exceeds $2^4 \cdot \sigma$ is less than $\operatorname{erfc}(\sqrt{2} \cdot 2^4) < 2^{-740}$, so we can set $B = 2^4 \cdot \sigma = 2^{21}$. This shows that only 22 queries are required to recover the noise of a ciphertext.

If the attacker has some additional knowledge about the implementation of the sampling algorithm used in the encryption software, it can do even slightly better. Assume, for example, that the implementation uses the Box–Muller transform to draw the noise using 32-bit numbers. Then one can show that the size of the noise will never be larger than $7 \cdot \sigma$, because the uniform variates that are input to Box–Muller cannot be arbitrarily close to 0. In this case the noise may be recovered using 21 queries.

3 Timing Side Channel on the Server

In the paper [CCCM22a] the authors claim that they can infer information about the noise of ciphertexts just by measuring the execution time of homomorphic gates, which, in this case, can be thought of as a simple addition of two ciphertexts. The claim is that since the ciphertexts depend on the noise, the execution time of the addition of two ciphertexts also depends on the noise and this dependence can be exploited in a so-called template attack. If this were true, it is easy to see how this might be useful in the above CVO attack, since it would allow narrowing down the range of the noise of a specific ciphertext by measuring the execution time of homomorphic addition. This is exactly what [CCCM22b] attempts to do. The result is that using techniques from [CCCM22a] they are able to recover the noise using 22 to 23 queries per ciphertext. This is quite underwhelming given that we already showed how to do that without the timing side channel. It suggests that there might be less information leakage than the authors claim. In fact, we claim that neither [CCCM22a] nor [CCCM22b] shows evidence of any information leakage on the server side whatsoever.

The high level idea in [CCCM22a] is to have the server mount a template attack: it builds a template by using a large number of ciphertexts it creates itself (and thus knows the noise) and measures the execution time of adding two ciphertexts. The resulting noise values are then clustered into buckets according to the measured execution time. This precomputation results in the template. During the online phase, when the server receives two ciphertexts, it adds them and measures the execution time. It then checks the bucket corresponding to the measured time and assumes the noise of the ciphertext will be in the same range; i.e., between the minimum and the maximum noise in the bucket. (The paper actually makes a distinction between positive and negative noise values, but this is irrelevant for our discussion here.)

The authors of [CCCM22b] provided some source code[3] and an example of a template they constructed that allows to reproduce their results. Figure 1 shows a depiction of the example template. Indeed, the noise is spread much further apart when the execution time is short, which might suggest that longer execution times allow to decduce a smaller noise range. But there are many more data points for the buckets with small execution time. So it seems the machine which this template was build on, ran on

[3] https://github.com/SEAL-IIT-KGP/CVO-TFHE

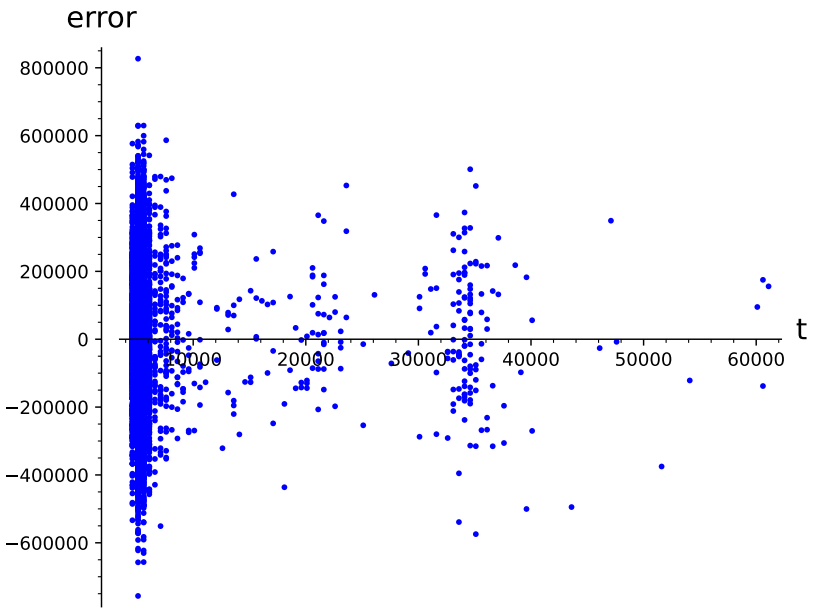

Figure 1: Plot of noise vs execution time in template

average relatively fast and occasionally took a little longer. This suggests a more natural explanation for the different noise ranges: the more often a Gaussian is sampled, the further the minimum and maximum of the samples will be apart. Intuitively, the more samples are drawn the more likely we will see outliers. So conditioning on the execution time does not actually change the marginal distribution, the template only reflects how often the execution time fell into a specific bucket.

We tested this hypothesis by simulating a template: we simply replaced each data point in each bucket of the given template with an i.i.d. sample from the (discrete) Gaussian with the appropriate (fixed) standard deviation and thus built a simulated template. The result is shown in Figure 2, which allows to verify at least visually that the simulated template has similar properties as the "real" template. To validate if this is indeed the case, we replaced the template in the authors' attack code by our simulated one and ran their attack. It turns out that our simulated template allowed recovering the key in the same amount of time.[4] If a template attack works with a simulated template that does not take any execution time measurements into account, we believe it is fair to say that the template is useless

[4]We had to slightly modify the code since the number of recovered noise values was hardcoded into the equation solver script and our template actually resulted in *more* recovered values, causing the script to crash.

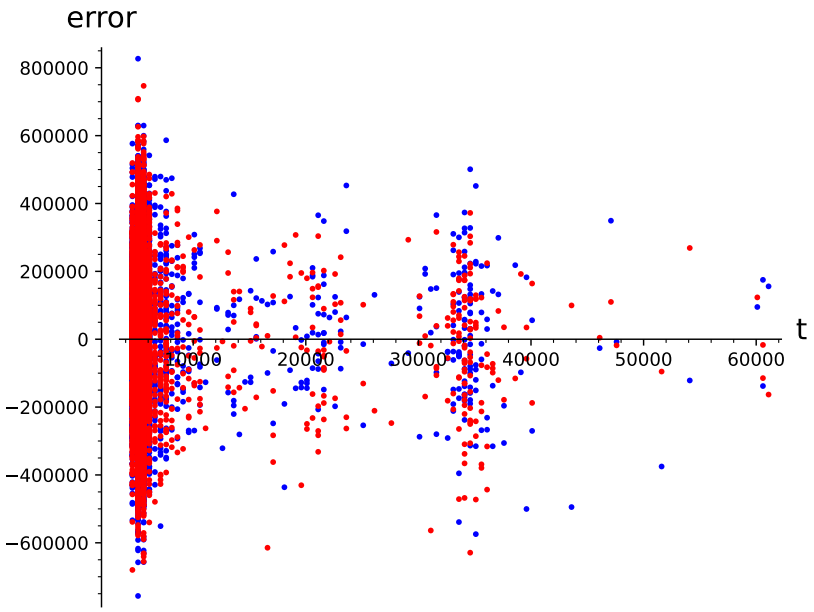

Figure 2: Plot of noise vs execution time in template (blue) and simulated template (red)

in the given attack and it does not demonstrate any leakage about the noise on the server side.

Discussion The authors of [CCCM22a] and [CCCM22b] claim that TFHE and FHEW might be weakend by side-channel analysis on the server side. But in both cases the server does not have any secret information and the attack does not even consider the bootstrapping keys. Note that TFHE and FHEW ciphertexts are proven IND-CPA secure under the LWE assumption (with different secret distributions). So claiming that TFHE and/or FHEW can be weakened using only public information (and not the bootstrapping keys), be it a template or any other side-channel attack on the server, is equivalent to claiming that the versions of LWE that TFHE and/or FHEW rely on are weak. This is because an adversary could simulate the machine being analyzed and mount such an attack. Given the current status of our understanding of LWE, this is an extraordinary claim.

References

[APS15] Martin R. Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of learning with errors. Cryptology ePrint

Archive, Report 2015/046, 2015. https://eprint.iacr.org/2015/046.

[CCCM22a] Bhuvnesh Chaturvedi, Anirban Chakraborty, Ayantika Chatterjee, and Debdeep Mukhopadhyay. Error leakage using timing channel in FHE ciphertexts from TFHE library. Cryptology ePrint Archive, Report 2022/685, 2022. https://eprint.iacr.org/2022/685.

[CCCM22b] Bhuvnesh Chaturvedi, Anirban Chakraborty, Ayantika Chatterjee, and Debdeep Mukhopadhyay. A practical full key recovery attack on TFHE and FHEW by inducing decryption errors. Cryptology ePrint Archive, Report 2022/1563, 2022. https://eprint.iacr.org/2022/1563.

[CGGI20] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. TFHE: Fast fully homomorphic encryption over the torus. *Journal of Cryptology*, 33(1):34–91, January 2020. doi:10.1007/s00145-019-09319-x.

[DM15] Léo Ducas and Daniele Micciancio. FHEW: Bootstrapping homomorphic encryption in less than a second. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 617–640, Sofia, Bulgaria, April 26–30, 2015. Springer, Heidelberg, Germany. doi:10.1007/978-3-662-46800-5_24.

[FHR21] Prastudy Fauzi, Martha Norberg Hovd, and Håvard Raddum. On the IND-CCA1 security of FHE schemes. Cryptology ePrint Archive, Report 2021/1624, 2021. https://eprint.iacr.org/2021/1624.

[FO99] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In Michael J. Wiener, editor, *Advances in Cryptology – CRYPTO'99*, volume 1666 of *Lecture Notes in Computer Science*, pages 537–554, Santa Barbara, CA, USA, August 15–19, 1999. Springer, Heidelberg, Germany. doi:10.1007/3-540-48405-1_34.