Work in Progress: Integrity Protection for Encrypted DNN Inference

Muhammad Santriaji[a], Gidhan Algary[b], Muhammad Fikriansyah[c], Rian Rajagede[a], Ardhi Yudha[a], Kyle Thomas[a], David Mohaisen[a] and Yan Solihin[a]

[a]University of Central Florida
[b]Universitas Dian Nuswantoro
[c]Universitas Telkom

Abstract.
Deep Neural Network (DNN) applications processing privacy-sensitive data must safeguard data integrity and confidentiality while ensuring high accuracy. While data confidentiality can be upheld through hardware-encrypted memory or software algorithms, DNN accuracy remains vulnerable to bitflip attacks, particularly within the ciphertext space. This extended abstract delves into the impact of bitflips in the ciphertext space when DNNs are encrypted via hardware-encrypted memory and the software algorithm, Fully Homomorphic Encryption (FHE). We further discuss potential strategies for preserving integrity.

1 Introduction

Deep Neural Networks (DNNs) are becoming a backbone for many applications because of their high accuracy. Many DNN applications run in a client-server environment where the data is collected in the client and then processed in the server. There are instances where server security might be untrusted, especially if the server is leased to other tenants. When working on confidential data like medical records, DNN must ensure the confidentiality of the inference.

Confidentiality can be protected using software and hardware. In hardware-encrypted memory like AMD SEV [1] and Intel SGX[3], the data and model of DNN are encrypted off the chip. Specialized hardware will only decrypt the data and models currently used for the computation to the on-chip memory. This ensures data confidentiality, safeguarding against cyberattacks.

Another approach to protecting inference confidentiality is Fully Homomorphic Encryption (FHE). Unlike the hardware approach, the encrypted data and models do not require decryption to be computed. The FHE approach is preferable when the server does not have the hardware capability to do a secure confidential computation.

However, both approaches suffer from bitflip attacks that can violate computation integrity and reduce the DNN accuracy. A recent study shows that running encrypted DNNs has a more significant risk to integrity. Recent works [13] show that the effect of bitflips on ciphertext space could lead to massive bit errors in plaintext, which aggravates the accuracy degradation effect.

Research in providing integrity protection for encrypted DNN has been scarce. Much research is focused on integrity protection in plaintext [15, 14, 9, 8, 6]. Meanwhile, the research in FHE encrypted DNN focus on accelerating the computation [10, 12, 4, 7].

Thus in this extended abstract, we will report our progress in protecting the integrity of encrypted DNN inference. In the first section, we will show the effect of bitflips in hardware-encrypted memory. Then we summarize our effort to protect the integrity of DNN inference in a hardware-encrypted memory. In the second section, we will report the effect of bitflips in DNN inference encrypted by FHE. Then we will discuss the potential solution to protect the integrity.

2 Threat Models

For this study, we focus on a specific scenario where a Deep Neural Network (DNN) is already trained, with its parameters fixed and immutable. The memory attacks of interest occur during the inference phase of the DNN, specifically within the ciphertext space. Under our threat model:

- Bit Alterations: We assume that the attacker possesses the capability to alter multiple bits in either the DNN model parameters or the DNN input data.
- Confidentiality Preservation: Despite the attacker's ability to change bits, it's essential to note that the actual data—its confidentiality—is not compromised and remains concealed from the attacker.
- Fault Detection Methodology: Our methodology for detecting these faults is inspired by and mirrors the approach detailed by Ponader et al. [13].

3 Privacy-preserving DNN Inference using Encrypted Memory

This section summarizes our finding in [16] by systematically studying bitflips in encrypted memory. We found that not all bits have the same importance in counter-mode encryption memory. In the floating-point datatype, we found that four bits reduce accuracy more significantly when flipped, and counter-intuitively, some of the less significant bits reduce accuracy more significantly than more significant bits when flipped.

3.1 Bitflip Impact on Encrypted DNN Inference

In the counter-mode encrypted memory (CMEM), the number and positions of bit errors in the ciphertext remain consistent in the plaintext.

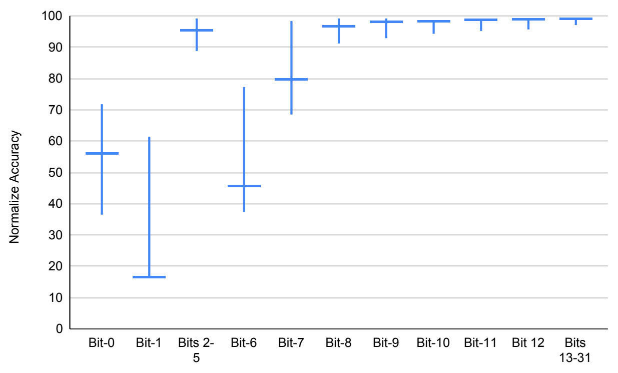

Figure 1. Effect of bitflips in term of accuracy for CMEM

We note that a unique characteristic of counter-mode encrypted memory (CMEM) is that the diffusion property of encryption only affects the pad generation, not the data stored in memory. To encrypt or decrypt, a pad is generated by encrypting or decrypting various components, including an address and a counter. Then the pad is XORed with data. Because of this, the diffusion property of encryption affects only counters. In contrast, for data conversion from plaintext to ciphertext, the number of bit flips and their locations do not change. From the attacker's point of view, CMEM allows them to precisely target any particular bits in DNN weights, even though they could also target bits of a counter to cause many bit errors. Thus, it is important to know what bits the attacker may find more attractive to flip, and they are affected by number representations.

Though some works have proposed narrower floating-point numbers, NN parameters are often stored as 32-bit floating-point or fixed-point representations. The IEEE 754 floating-point format, consisting of a 1-bit sign (s; bit 0), 9-bit exponent (e; bits 1-9), and 23-bit mantissa (m; bits 10-32). These components represent a number using the following formula: $F = (-1)^s \times 2^{(e-b)} \times (m+1)$, where $b = 2^{(k-1)} - 1$ and k is being the number of bits used for the exponent representation. Furthermore, the exponent bits are divided into a 1-bit sign (bit 1) and 8-bit magnitude (bits 2 to 9).

3.2 *Characterization of Bit Flip Impacts on Accuracy*

We start by choosing a particular bit in the floating-point representation. Then, for the target network, we randomly choose a parameter (in any layer) and flip the particular bit of that parameter. Then we measure the accuracy of the network's classification output for the network's testing inputs. We repeat the experiment over 1,000 times, randomly selecting a parameter to inject the bit flip. Then, we collect the minimum accuracy (*Low*), average accuracy (*Average*), and maximum accuracy (*High*). We test our experiment on an MNIST-trained neural network with 34,720 weights trained. The results are shown in Figure1. The figure shows the accuracy degradation in terms of each bit position's high, average (vertical line in the middle), and low.

We note the following interesting observations. First, the table shows four notable bits that reduce the network accuracy to below 80%: bits 0, 1, 6, and 7. All other bits do not reduce accuracy by more than 5%. Furthermore, when flipped, none of the mantissa bits greatly reduce the accuracy.

Second, bit 1 turns out to have the most considerable impact on accuracy, reducing it from 98% to 16.5% with a single bit flip, showing the severe impact a single bit out of a million can have.

From the representation, bit 1 essentially represents the sign for the exponent. Hence flipping the bit changes a number from very large to very small, or vice versa while having the same sign, i.e., a positive number remains positive, and a negative remains negative.

Third, bit 0 changes the sign of the number. It also substantially affects accuracy, but not as much bit 1. From here, we can conclude that changing a number magnitude significantly reduces accuracy more than changing its sign.

Fourth, surprisingly, the four most significant exponent bits do not significantly affect accuracy. Instead, the fifth and sixth most significant exponent bits (bits 6-7) reduce accuracy much more than other exponent bits, even though the more significant the exponent bit, the more it changes the magnitude of the number. Investigating this further, we found that the activation function is the culprit.

4 Privacy-preserving DNN Inference using Fully Homomorphic Encryption

This section studies bitflips attacks in DNN inference encrypted by FHE. FHE is an advanced cryptographic technique that allows computations to be performed on encrypted data without requiring decryption. It enables data to be encrypted in a way that retains its arithmetic properties, allowing mathematical operations to be applied directly to the encrypted data. This means that even with encrypted data, complex computations can be executed, and the final result will also be in encrypted form. It ensures the confidentiality of sensitive data throughout the entire computation process.

4.1 *Bitflip Impact*

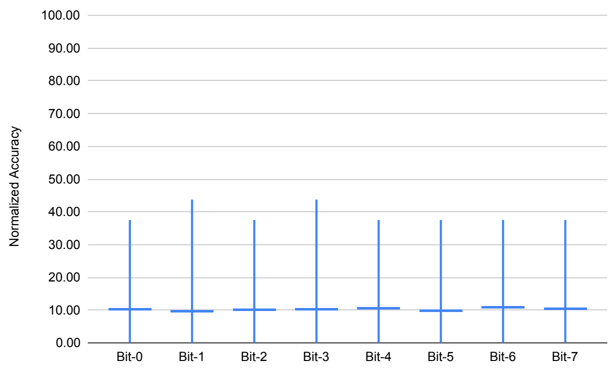

Figure 2. Effect of bitflips in term of accuracy for FHE-Cryptonet

Figure 2 shows the effect of bit-flip in multiple bits of MNIST digit classification dataset in Cryptonet [5] encrypted by FHE. Cryptonet is a DNN architecture explicitly designed to do encrypted inference using FHE. In each of the 5000 iterations, we randomly applied a bit-flip to one of the data input bits. From these results, we can make several observations. First, the average accuracies are in the 10s%, which means a single random bit-flip attack in FHE has a greater detrimental impact compared to a similar bit-flip attack in counter-mode encryption from the previous section. Second, the resulting accuracies show an extensive range, from zero percent accuracies to

Algorithm 1 Layer Hashing — Detection Stage

```
 1: procedure LAYER HASHING DETECTION
 2:     $N \leftarrow NetworkSize$
 3:     $SNH \leftarrow StoredNetworkHash$
 4:     $SLH \leftarrow StoredLayerHashess$
 5:     $HN \leftarrow NewNetworkHash$
 6:     if $HN == SLH_i$ then
 7:         return No Errors
 8:     end if
 9:     for $L = 1, 2, \ldots, N$ do
10:         $HL, HO \leftarrow NewHash, SLH_L$
11:         if $HL \;= HO$ then
12:             Report Error Layer
13:         end if
14:     end for
15: end procedure
```

40%. Our results suggest that encrypted DNN inference using FHE is more susceptible to bit-flip attacks than counter-mode encryption.

5 Potential Solution

To effectively detect both malicious bit-flip attacks and unintentional bit flips, we delve into techniques that identify memory errors impacting NN parameters. The detection subsection summarizes our findings from [16] for the hash method. We also offer a weighted checksum solution for encrypted DNN using FHE in [11]. Then we explore the techniques to provide DNN robustness. We found that the current state of the art cannot protect the encrypted DNN using FHE against a single random bit-flip.

5.1 *Hash Detection Method*

Algorithm 1 describes our solution to detect bitflip using layer hashing. Hashing is a common method for ensuring data integrity in various applications. A hash function maps an input string, denoted as M, of arbitrary length to an output string $h(M)$ of a fixed bit-length d. This output string is called the "message digest," and any change to M changes the digest. Our layer hashing method (LHM) has a 100% detection rate for all cases. There is no way to evade LHM detection, even through layer permutation.

LHM has two drawbacks. First, any change in a parameter, even small ones with small magnitudes, requires the hash to be recalculated. Second, LHM cannot pinpoint erroneous parameters. Hence recovering from an error requires restoring the entire layer.

5.2 *Robustness Method*

In this experiment, we perform the experiment on ResNet-18 trained with the CIFAR-10 dataset. We encrypt the input data with FHE and then apply a single bitflip randomly. Then we decrypt the data into plaintext and send it as input to the DNN to simulate the effect of bitflip in ciphertext space. Figure 3 shows the effect of a single bitflip in encrypted data input by FHE on the DNN accuracy. We can see that the current state-of-the-art, the robust-trained DNN model and control-based model [2] cannot prevent accuracy degradation. The methods give about 10% accuracy in ResNet inferring a CIFAR-10 dataset. This happens because a robust DNN model is designed to withstand adversarial attacks and maintain accurate predictions even when exposed to small, intentional input perturbations. However, the single bitflip attack in the ciphertext would become multiple bitflips in plaintext, making the robust model ineffective.

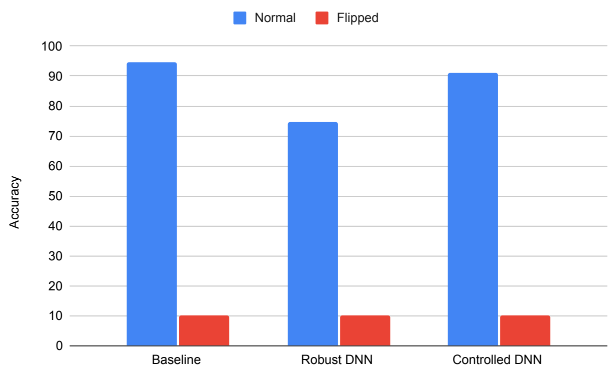

Figure 3. Effect of bitflips in terms of accuracy for multiple robustness protection

6 Conclusion and Future Works

In this extended abstract, we discussed our advancements in ensuring integrity protection for encrypted DNN inference. While we've made strides in detecting bitflip anomalies, addressing the resulting inference inaccuracies remains a challenge. Currently, the sole recourse is to recompute the entire DNN, which can be resource-intensive. Moving forward, our aim is to develop a 'self-healing' mechanism for encrypted DNNs, aiming to mitigate these inaccuracies without the need for full recomputation, thereby reducing computational costs

7 Acknowledgement

This work was partially supported by the Ministry of Education, Culture, Research, and Technology of the Republic of Indonesia through the Garuda Ace Project, with Project IDs 21-I1 and 24-I4.

References

[1] AMD, 'AMD SEV-SNP: Strengthening VM isolation with integrity protection and more', *White Paper*, (Feb 2020).

[2] Zhuotong Chen, Qianxiao Li, and Zheng Zhang, 'Self-healing robust neural networks via closed-loop control', *J. Mach. Learn. Res.*, **23**(1), (jan 2022).

[3] Intel Corporation, 'Intel architecture memory encryption technologies specification', *Ref: #336907-002US*, (2019).

[4] Bo Feng, Qian Lou, Lei Jiang, and Geoffrey Fox, 'CRYPTOGRU: Low latency privacy-preserving text analysis with GRU', in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 2052–2057, Online and Punta Cana, Dominican Republic, (November 2021). Association for Computational Linguistics.

[5] Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing, 'Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy', in *International conference on machine learning*, pp. 201–210. PMLR, (2016).

[6] Yanan Guo, Liang Liu, Yueqiang Cheng, Youtao Zhang, and Jun Yang, 'Modelshield: A generic and portable framework extension for defending bit-flip based adversarial weight attacks', in *2021 IEEE 39th International Conference on Computer Design (ICCD)*, pp. 559–562, (2021).

[7] Chiraag Juvekar, Vinod Vaikuntanathan, and Anantha Chandrakasan, 'Gazelle: A low latency framework for secure neural network inference', in *Proceedings of the 27th USENIX Conference on Security Symposium*, SEC'18, p. 1651–1668, USA, (2018). USENIX Association.

[8] Jingtao Li, Adnan Siraj Rakin, Zhezhi He, Deliang Fan, and Chaitali Chakrabarti, 'Radar: Run-time adversarial weight attack detection and accuracy recovery', *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 790–795, (2021).

[9] Jingtao Li, Adnan Siraj Rakin, Yan Xiong, Liangliang Chang, Zhezhi He, Deliang Fan, and Chaitali Chakrabarti, 'Defending bit-flip attack through dnn weight reconstruction', in *Proceedings of the 57th ACM/EDAC/IEEE Design Automation Conference*, DAC '20. IEEE Press, (2020).

[10] Qian Lou and Lei Jiang, 'She: A fast and accurate deep neural network for encrypted data', in *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., (2019).

[11] Qian Lou, Muhammad Santriaji, Ardhi Wiratama Baskara Yudha, Jiaqi Xue, and Yan Solihin. vfhe: Verifiable fully homomorphic encryption with blind hash, 2023.

[12] Qian Lou, Yilin Shen, Hongxia Jin, and Lei Jiang, 'Safenet: A secure, accurate and fast neural network inference', in *International Conference on Learning Representations*, (2020).

[13] Jonathan Ponader, Kyle Thomas, S. Kundu, and Y. Solihin, 'Milr: Mathematically induced layer recovery for plaintext space error correction of cnns', *2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 75–87, (2021).

[14] Adnan Siraj Rakin, Zhezhi He, and Deliang Fan, 'Bit-flip attack: Crushing neural network with progressive bit search', *2019 IEEE/CVF International Conference on Computer Vision (ICCV '19)*, 1211–1220, (Apr 2019).

[15] Adnan Siraj Rakin, Zhezhi He, Jingtao Li, Fan Yao, Chaitali Chakrabarti, and Deliang Fan, 'T-BFA: targeted bit-flip adversarial weight attack', *CoRR*, **abs/2007.12336**, (2020).

[16] Kyle Thomas, Muhammad Santriaji, David Mohaisen, and Yan Solihin, 'Exploration of bitflip's effect on dnn accuracy in plaintext and ciphertext', *IEEE Micro*, 1–11, (2023).