Remote Rowhammer Attack using Adversarial Observations on Federated Learning Clients

Jinsheng Yuan*, *Student Member, IEEE,* Yuhang Hao, Yun Wu, *Member, IEEE,* Weisi Guo, *Senior Member, IEEE,* Chongyan Gu, *Senior Member, IEEE*

***Abstract*—Federated Learning (FL) has the potential for simultaneous global learning amongst a large number of parallel agents, enabling emerging AI such as LLMs to be trained across demographically diverse data. Central to this being efficient is the ability for FL to perform sparse gradient updates and remote direct memory access at the central server. Most of the research in FL security focuses on protecting data privacy at the edge client or in the communication channels between the client and server. Client-facing attacks on the server are less well investigated as the assumption is that a large collective of clients offer resilience.**

Here, we show that by attacking certain clients that lead to a high frequency repetitive memory update in the server, we can remote initiate a rowhammer attack on the server memory. For the first time, we do not need backdoor access to the server, and a reinforcement learning (RL) attacker can learn how to maximize server repetitive memory updates by manipulating the client's sensor observation. The consequence of the remote rowhammer attack is that we are able to achieve bit flips, which can corrupt the server memory. We demonstrate the feasibility of our attack using a large-scale FL automatic speech recognition (ASR) systems with sparse updates, our adversarial attacking agent can achieve around 70% repeated update rate (RUR) in the targeted server model, effectively inducing bit flips on server DRAM. The security implications are that can cause disruptions to learning or may inadvertently cause elevated privilege. This paves the way for further research on practical mitigation strategies in FL and hardware design.

***Index Terms*—Federated Learning, Rowhammer Attack, Fault Injection, Reinforcement Learning**

I. Introduction

A. Motivation

Recent advances in artificial intelligence (AI), particularly in federated learning (FL), have significantly enhanced privacy preservation by enabling distributed model training across numerous client devices without sharing raw data [1]. However, the increasing complexity and scale of these distributed AI systems also expose them to novel and sophisticated threat vectors that transcend traditional adversarial machine learning techniques [2]. Among hardware security attacks, the Rowhammer attack stands out due to its potency in inducing bit-flips within Dynamic Random Access Memory (DRAM) through targeted electromagnetic interference from frequent activations of adjacent rows [3]. Despite various industry-standard countermeasures, such as Target Row Refresh (TRR), recent studies highlight persistent risks, demonstrating that modern DRAM modules remain vulnerable to advanced Rowhammer attacks [4], [5].

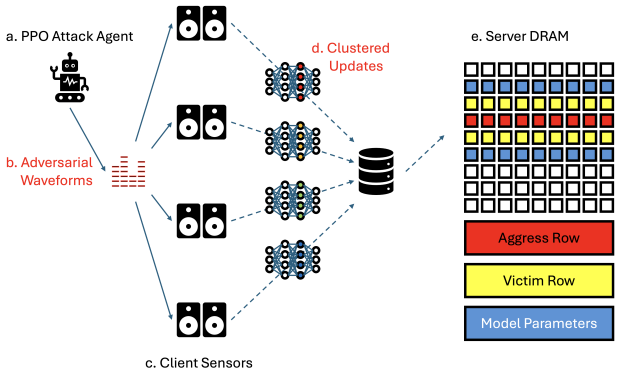

Fig. 1. Framework of our proposed threat vector. a) a PPO agent generates b) adversarial waveforms that interfere the inputs of c) client sensors of FL, resulting the clients to send d) clustered updates, which could trigger Rowhammer attack on e) server DRAM

Motivated by the following complementary observations, we explore the potential of combining physical adversarial techniques with reinforcement learning (RL) methods to orchestrate Rowhammer bit-flips remotely in FL environments. On the one hand, existing Rowhammer attack strategies typically presume direct or indirect software-level access to victim hardware [6]–[8]. This assumption limits their practical applicability, particularly in scenarios where adversaries have no explicit system-level privileges or direct remote memory access. While on the other, physical adversarial attacks have demonstrated the feasibility of using carefully crafted electromagnetic or acoustic perturbations to deceive AI-driven sensor inputs without requiring direct digital intrusion [9], [10].

Specifically, modern FL systems deployed at scale frequently integrate efficiency optimizations, such as sparse gradient updates, pinned-memory regions, huge pages, and Remote Direct Memory Access (RDMA), which significantly increase their vulnerability to carefully crafted physical perturbations. We hypothesize that these optimizations, while beneficial for performance, inadvertently facilitate indirect Rowhammer attacks by enabling precisely controlled memory access patterns favorable to inducing DRAM bit-flips.

This work was funded by the EPSRC New Investigator Award (EP/X009602/1)

Jinsheng Yuan and Weisi Guo are with the Faculty of Engineering & Applied Sciences, Cranfield University, UK. (e-mail: Jinsheng.Yuan@cranfield.ac.uk, Weisi.Guo@cranfield.ac.uk)

Yuhang Hao, Yun Wu and Chongyan Gu are with Queen's University Belfast. (e-mail: yhao09@qub.ac.uk, yun.wu@qub.ac.uk, C.Gu@qub.ac.uk)

*Corresponding author.

Consequently, an adversary could exploit federated clients by interfering their sensor inputs with maliciously crafted noises (e.g., acoustic or electromagnetic signals) to indirectly manipulate server-side memory access patterns, ultimately causing disruption and degradation of FL system performance without direct access to victim hardware.

B. Contributions

In this paper, we propose a novel threat vector that leverages a reinforcement learning-based Proximal Policy Optimization (PPO) agent to induce Rowhammer bit-flips on federated learning servers indirectly via physical interference at client devices. The PPO agent generates stealthy adversarial waveforms projected into client sensors (e.g., microphones), triggering clustered model parameter updates on the server. These updates, amplified through efficiency optimizations such as RDMA, pinned memory, and sparse gradient updates, achieve the activation frequency and regularity necessary to induce Rowhammer attacks on server-side DRAM.

The key contributions of this work include:

1) **Novel Attack Vector**: We introduce the first physical-domain-driven Rowhammer attack vector specifically targets federated learning (FL) systems. Unlike traditional Rowhammer attacks, our approach does not require direct access or explicit control over victim hardware or software resources, greatly expanding its potential attack surface.
2) **PPO-based Attack Generation Framework**: We design a two-stage reinforcement learning framework that generates stealthy adversarial waveforms, ensuring sustained and clustered parameter updates within FL models, which consequently induce Rowhammer bit-flips.
3) **Vulnerability Analysis**: We perform experimental evaluations using realistic federated learning scenarios with popular automatic speech recognition (ASR) models and datasets. Our experiments demonstrate that Rowhammer bit-flips can be induced on FL systems enhanced with common efficiency optimizations through adversarial interference on client sensors.
4) **Potential Mitigation Strategies**: We discuss practical countermeasures and security recommendations to defend against the proposed threat vector, such as adversarial input detection and systematic analysis of optimization technique deployment.

C. Organization

The remainder of this paper is organized as follows. Section II provides a relevant background on DRAM architecture, Rowhammer vulnerabilities, FL security, and physical adversarial attacks. Section III formally describes the proposed attack vector, including motivation, feasibility analysis, and attack formulation. Section IV details the experimental setups, datasets, and evaluation methodology. Section V presents extensive experimental results, analyzing the effectiveness of the PPO-based adversarial attack framework. Section VI discusses potential mitigation techniques and recommendations to counteract the proposed threat. Finally, Section VII concludes the paper and outlines directions for future research.

II. Background and Related Work

A. DRAM Architecture and Rowhammer Attack

Modern DRAM modules can be classified into single-rank, dual-rank, and multiple-rank according to the number of ranks. Each rank includes multiple chips that share the same data bus. Each chip contains multiple banks, which are organized into bank groups. It is necessary to identify the bank group number when accessing one specific bank. A DRAM bank consists of an array of memory cells, where all cells in a horizontal direction are collectively referred to a DRAM row, while those in a vertical direction are referred to a DRAM column. Each cell represents one logical bit. Depending on the memory strategy, a fully charged cell indicating a logical '1' is called a true-cell, whereas its counterpart, the anti-cell, denotes a logical '0'. The specific DRAM organization is shown in Fig. 2. When data needs to be read from or written to DRAM, the memory controller (MC) issues an `ACT` command to open a row. The entire row of data is activated by the wordline and transferred to the sense amplifiers (also known as the row buffer). The bitlines then access specific columns within the row buffer to perform the read or write operation. Finally, the MC issues a `PRE` command to precharge the bitlines and close the currently active row.

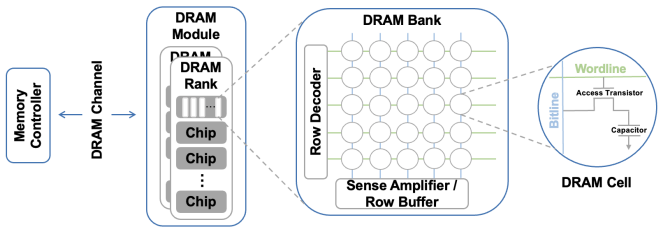

Fig. 2. Hierarchical organization of a DRAM system, depicting the structural composition from the memory controller and channel architecture through modules, ranks, and chips, down to individual bank-level components (row decoder, sense amplifier/row buffer) and the underlying DRAM cell structure (wordline, access transistor, capacitor). The schematic illustrates the multi-tiered interconnectivity and functional partitioning critical to DRAM operation, spanning system-level interfaces to transistor-level charge storage mechanisms.

For DRAM access, whether it is a read or write operation, the MC issues both an `ACT` command and a `PRE` command for the accessed row. The fundamental cause of the Rowhammer attack lies in the frequent issuance of `ACT` commands. When the adjacent rows (aggressor rows) of a victim row are repeatedly activated, the wordline voltage, which drives the aggressor row, changes rapidly and frequently. It then induces electromagnetic coupling with the memory cells in the victim row, altering their charge state. It should be noted that even if only a single aggressor row is used for the Rowhammer attack, it still requires the coordinated activation of two rows within the same bank. This is because if only one aggressor row is accessed repeatedly, the MC will not precharge this row before completing all operations, making the attack ineffective.

Furthermore, the reason why the Rowhammer attack requires rapid and frequent access to the aggressor row lies in the DRAM refresh mechanism. DRAM performs periodic refresh operations to maintain the charge state of all memory cells. Typically, this refresh period is 64 ms or 128 ms, but under high-temperature conditions, it could be reduced to 32 ms. In each refresh period, the MC typically issues 8192 `REF` commands, which sequentially refresh all rows on the DRAM module. Taking a 64 ms refresh period as an example, the average interval between each `REF` command is 7.8 μs. If the number of activations on the aggressor row fails to effectively cause charge leakage in the victim row before its next refresh, the attacker must accumulate more activations again in next refresh period to execute Rowhammer attack.

The factors influencing the Rowhammer attack include manufacturing, data pattern, and access pattern. Limited by the current manufacturing process, there are some inherently vulnerable memory cells in each DRAM chips that are more prone to charge leakage compared to normal cells, regardless of other factors. The data pattern refers to the scenario where the Rowhammer attack is more likely to succeed when the charge states of memory cell in the aggressor row and victim row are different [11]. Although adversary attempting to tailor the data in aggressor row based on the victim data may be affected by anti-cells, existing research has proved that the proportion of anti-cells with the same chip is very small [3]. The access pattern includes single-sided, double-sided, and many-rows Rowhammer attack [4], [12]. The double-sided Rowhammer, compared to a single-sided, involves two aggressor rows targeting the same victim row. This type of attack is generally more effective than single-sided Rowhammer. However, it requires more detailed memory mapping preparation. On the other hand, the many-rows Rowhammer is often used to bypass the TRR protection mechanism.

```
1       loop:
2           mov (X), %eax
3           mov (Y), %ebx
4           clflushopt (X)
5           clflushopt (Y)
6           mfence
7           jmp loop
```

Listing 1. Pseudo code of Rowhammer attack utilising the `clflushopt` instruction

The key points and progress details of executing and exploiting a Rowhammer attack are as follows:

1) **Selecting Rowhammer access patterns:** The specific DRAM modules of target system guides the choice of Rowhammer attack access patterns, primarily depending on whether the victim chips have TRR enabled. For early DDR3, both simple single-sided and more effective double-sided Rowhammer are ideal access patterns. However, for modern DDR4 chips, where TRR is widely implemented, many-rows Rowhammer and its variants must be used instead [4], [13]. This is because TRR protection mechanism rely on its internal sampler [14]. By using multiple dummy rows, attackers can successfully overload the size of sampler, preventing the actual aggressor rows from being affected by extra refresh operations. It is worth noting that error-correcting code (ECC) has been proven ineffective against this attack [15], as efficient access patterns such as double-sided Rowhammer can also overwhelm ECC mechanisms [16].
2) **Determining Address Mapping:** Since Rowhammer attack is triggered by adjacent rows of victim data, determining the addresses of aggressor rows is crucial. However, the virtual memory address used in application do not correspond to actual DRAM address. The process of locating the correct address typically involves two mapping stages [17]. The first is application-level virtual address to physical address. In operating system, the memory management unit (MMU) in the CPU, maps contiguous virtual addresses to non-contiguous physical addresses in a non-linear manner. In some operating systems, such as Ubuntu, the mapping can be retrieved via `/proc/pid/pagemap`. The next stage involves physical address to DRAM address mapping, which varies by CPU architecture and DRAM manufactory. This mapping is undocumented, and common approaches to reverse-engineering it rely on side-channel analysis [5], [18]–[20]. A key point to mention is that in a double-sided Rowhammer scenario, a common strategy for finding usable contiguous physical addresses is to use huge pages.
3) **Avoiding Cache Effects:** Frequently accessed data will be stored in the cache to avoid CPU complex operations. Thus, Rowhammer attack requires to use specialized CPU instructions to bypass cache to achieve high-frequency activations of aggressor rows in real attack scenarios. Currently, the main instructions used to achieve this purpose include `clflush` [3], [21], `clflushopt` [22], and `invd` or `wbinvd` [17]. In a carefully crafted attack sequence, incorporating the aforementioned instructions can help the attacker bypass the cache. Sometimes, this also requires the use of memory barriers such as `mfence`, `lfence`, and `sfence`, but they may not achieve the optimal activation rate. Listing 1 illustrates a Rowhammer attack sequence utilising the `clflushopt` instruction and memory barrier. Additionally, there are some attacks based on cache line conflicts to bypass [15], [23].
4) **Ensuring Effective Activation Rate and Count:** After determine all above conditions, attackers should ensure enough activations to induce bit-flips. There are two key factors to ensure the successful execution of the attack. One is maintaining an effective activation rate. Even though the standard tRC (Row Cycle Time) is only 46 ns, the average interval between two activation instructions should be around 220 ns, considering the influence of other instructions. The other factor is accumulating a sufficient number of activations to ensure that the aggressor rows are activated enough times to induce bit flips. Additionally, when TRR are present, the dummy rows used to bypass TRR must also be accounted for in the total activation count.

The total number of activations should not exceed the maximum allowable activations within a single refresh period (64 ms) to avoid DRAM refresh interventions.

B. Federated Learning

Federated Learning (FL) is a distributed machine learning paradigm that enables collaborative model training across multiple devices while retaining the raw data locally [1]. Unlike conventional centralized training schemes, FL enhances data privacy by transmitting only model updates, e.g., gradients, to a central server for aggregation. This distributed framework is particularly advantageous for use cases where data confidentiality and regulatory compliance are paramount (e.g., healthcare, finance, or mobile devices). Moreover, the scalability of FL allows diverse and expansive datasets to be leveraged by increasing the number of participating clients, thereby improving generalization across heterogeneous data distributions. The typical FL process involves the following steps:

1) **Local Training:** Each client trains a local model using its own private dataset.
2) **Model Aggregation:** In each communication round, a designated subset of clients transmits model updates (e.g., weights or gradients) to the central server. These updates are then aggregated into a global model using algorithms such as FedAvg [1].
3) **Redistribution:** The aggregated global model, depending on the training strategy, is redistributed to the participating clients, or all clients, which then continue local training in the next round.

1) Security Threats in Federated Learning: Federated learning (FL) systems are vulnerable to a variety of security threats that arise from both adversarial machine learning (ML) techniques and conventional communication/network-based attacks. The former category includes model poisoning, backdoor, and inference attacks, while the latter mainly consists of eavesdropping and distributed denial-of-service (DDoS) attacks. In this paper, we focus on an attack vector more closely aligned with adversarial ML methodologies.

- **Poisoning Attacks:** Poisoning attacks are a specialized form of injection attack that targets ML models. Within FL, adversaries exploit malicious or compromised clients to manipulate the training process, which can degrade the global model's performance. Specifically, *data poisoning* involves injecting or modifying training samples (often through mislabeled or corrupted data), whereas *model poisoning* targets model updates (e.g., gradients or weights) to undermine accuracy or embed hidden behaviors.
- **Backdoor Attacks:** Backdoor attacks implant triggers in an ML model during training, allowing the adversary to covertly control the model output. By submitting local updates with deliberately engineered triggers or patterns, malicious clients can force the global model to behave normally for most inputs, yet generate attacker-controlled outputs for specific trigger inputs. These attacks are especially dangerous because they usually go undetected until the trigger is activated.
- **Inference Attacks:** Inference attacks exploit partial information shared in FL systems to reveal or deduce private data from participating clients. By analyzing the distributed global model, adversaries may ascertain membership (i.e., whether a particular sample was part of the training set) or other sensitive attributes. This highlights the necessity for more robust privacy-preserving techniques in FL.

2) Efficiency Optimization: Federated Learning (FL) systems, especially those deployed in high-performance computing (HPC) facilities and data centers, often incorporate efficiency-oriented techniques such as sparse gradient updates, page-locked memory, and Remote Direct Memory Access (RDMA) for accelerated communication and reduced overhead.

- **Sparse updates** in federated learning reduce communication overhead by transmitting only a subset of model parameter changes that exceed a chosen threshold or rank among the largest by magnitude. Clients compute local gradients and select the most significant updates to send to the server which aggregates these sparse contributions to refine the global model. This approach can significantly lower bandwidth requirements and accelerates training rounds, especially for federations of large number of clients.
- **Page-lock memory**, also referred to as pinned-memory, marks a region of memory as 'pinned', keeping the section of memory resident in RAM at all times, hence improves the efficiency of program by avoiding delays caused by relocation or page-out of memory. In modern GPU based deep learning scenario, pinned-memory locks a section of DRAM to enable asynchronous transfers between DRAM and VRAM on GPUs, which significantly improves efficiency in data loading and training throughput.
- **Huge Pages** are a memory management technique that allows the operating system to use larger page sizes than the default 4KB, reducing the number of page table entries, which can improve translation lookaside buffer (TLB) hit rates and reduce the overhead of memory management. This can lead to performance improvements in applications that have large memory footprints and high memory access rates, such as high-performance computing (HPC) applications. For AI workloads, variables such as model parameters, data are often too large for default page size, hence huge pages can significantly improve the performance of AI workloads.
- **DMA and RDMA** Direct Memory Access (DMA) is a feature of computer systems that allows certain hardware subsystems to access main system memory (RAM) independently of the central processing unit (CPU). Remote Direct Memory Access (RDMA) is a specialized form of DMA that enables high-speed data transfers between networked computers by allowing data to be transferred directly from the memory of one computer to the mem-

ory of another without involving the CPU. RDMA is commonly used in high-performance computing (HPC) and data center environments to accelerate data transfers and reduce latency. By bypassing the CPU and operating system, RDMA can achieve latencies as low as 1-2 μs and bandwidths exceeding 200 Gbps, making it ideal for applications that require high-speed, low-latency data transfers, such as machine learning training and distributed computing. In FL systems, RDMA can significantly reduce communication overhead and improve scalability by enabling efficient aggregation and distribution of model parameters between central servers and distributed clients. By minimizing synchronization delays and network bottlenecks, RDMA can enhance the overall training performance and accelerate convergence in FL systems, particularly in large-scale deployments involving hundreds or thousands of clients.

C. Physical Adversarial Attacks

Deep learning has revolutionized a wide range of applications, extending from computer vision to audio processing, enabling technologies such as autonomous driving, surveillance, biometric authentication, robotics, and voice-controlled systems. However, the reliability and robustness of these systems have been called into question due to adversarial attacks. These attacks involve purposefully crafted perturbations, which are often imperceptible to human observers, that can mislead or corrupt a model's predictions. Originally explored in the digital domain, adversarial examples have since transitioned into the physical world, maintaining their deceptive properties even after real-world transformations like printing, photographing, or acoustic playback [24]–[26].

In the context of computer vision, early works by Szegedy et al. [24] and Goodfellow et al. [25] unveiled the vulnerability of neural networks to subtle pixel-level perturbations. Kurakin et al. [26] and Eykholt et al. [9] extended these findings into the physical domain, revealing that adversarial examples remain effective when transformed into physical objects or printed images. Consequently, research has demonstrated that adversarial patches [27], malicious alterations to wearable items, and tampered traffic signs [28] can consistently deceive image classification and object detection models. Such attacks often operate under the radar of human vision and can be engineered to withstand environmental noise and viewpoint changes [29]–[31].

Adversarial attacks have also become common in the audio domain. Speech recognition and voice-activated systems are increasingly critical in applications ranging from smart home devices to security systems. Adversarial perturbations can be embedded in audio signals to cause automatic speech recognition (ASR) models to misinterpret commands or produce incorrect transcriptions. These perturbations can be made so subtle that they remain inaudible to humans. For instance, Carlini and Wagner [10] demonstrated targeted adversarial examples against speech-to-text systems, while Yuan et al. [32] and Qin et al. [33] introduced methods for embedding covert commands within naturalistic audio clips. As with visual adversaries, these audio-based attacks persist in the physical domain, maintaining their effectiveness when played through speakers and recorded by microphones, thus bypassing human scrutiny and threatening the security of voice-controlled systems.

III. Attack Vector Formulation

A. Inspiration

In high-performance computing (HPC) environments, optimizations meant to enhance efficiency can inadvertently introduce vulnerabilities exploitable by Rowhammer-style attacks. We identify three specific properties induced by optimizations commonly found in federated learning (FL) systems, particularly suited for exploitation:

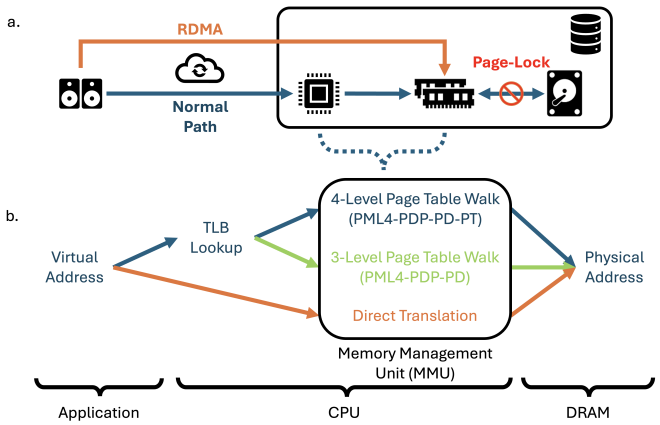

Fig. 3. DRAM access pathways and processes of FL. a) DRAM access pathways. For normal access, client updates go through network to the CPU of server before getting to the DRAM of the server, while RDMA allows clients to bypass the CPU; b) DRAM access processes. Normally, when mapping virtual addresses of applications to physical addresses on DRAM, the CPU lookup TLB, and walk through the page table if TLB miss. Using Huge Page reduces a level in page table walk, and DMA/RDMA bypass both TLB lookup and page table walk. Page-lock prevents variables from offloading from DRAM to disk.

1) **Stability in Physical Address:** In FL workflows, training data and model parameters are assigned to locked huge pages to maximize data transfer efficiency between CPU and GPU. By allocating the buffer as a huge page, fewer TLB entries are needed, reducing address-translation overhead, while page-locking guarantees that the memory cannot be paged out or moved. DMA/RDMA then enables the device to read or write directly to this pinned, contiguous memory region. When employing these three techniques simultaneously, the operating system ensures that the allocated memory region remains both physically contiguous and non-swappable, effectively 'locking in' a stable translation to a fixed physical address, especially when considering a short period of time such as the typical refresh period of 64ms window for typical rowhammer attacks. Therefore, potential attacker can lock the target row via the FL workflows, e.g., locking a physical DRAM row via specific continuous parameters of the global model.

2) **Minimal Cache Interference:** FL systems that employ DMA or RDMA and non-temporal memory accesses to optimize distributed parameter exchange enable more frequent and predictable direct DRAM accesses. RDMA's zero-copy semantics let attackers initiate high-frequency remote memory reads/writes that bypass CPU caches on a targeted node's DRAM. Additionally, sparse updates—commonly used in FL to reduce bandwidth—focus access patterns on a subset of model parameters. By continuously hammering these frequently accessed parameters or their associated pinned-memory regions, attackers can ensure sustained direct DRAM access while minimizing cache interference. Non-temporal instructions, often integrated for performance reasons, further reduce cache pollution, promoting repeated DRAM row activations.
3) **Sustained High Rate of Row Activations:** With the reduced latency and jitter in memory accesses by above techniques, potential attackers can achieve high hammering frequencies through the large number of clients with sparse gradient updates. Specifically, the gather-scatter mechanism, commonly used to efficiently aggregate sparse gradient updates by collecting and redistributing only the most significant gradients (e.g., the top 0.05%), further minimizes data transfer overhead. This concentration on a small subset of parameters not only enhances communication efficiency but also maintains consistent memory access patterns, reinforcing the repeated activation of targeted memory rows required for effective Rowhammer exploitation.

B. Evaluation Metrics

1) Notations: To evaluate and validate the proposed attack strategy, we map the repeated rate of sparse client updates to the number of Rowhammer operations required for DRAM. We define the following terms:

- $H_{\text{max}}(p)$: The maximum number of sparse updates processed within a single DRAM refresh period for an update fraction p that leads to update size $S_{\text{update}}(p)$.
- p: The fraction of model parameters updated in each sparse update.
- $T_{\text{flip}}(r)$: The Rowhammer threshold, measured as the number of updates required to induce a bit-flip in row r.
- $\overline{T_{\text{flip}}}$: The average Rowhammer threshold across all rows.

2) RL Performance Metrics: For the RL agent, we define the following metrics that comprehensively quantify the clustering behavior and consistency of induced parameter updates:

- **Repeated Update Rate (RUR)**: Measures consistency of parameter indices updated across episodes. Higher RUR indicates persistent, targeted updates:

$$\text{RUR} = \frac{\sum_{t=1}^{T-1} |U_t \cap U_{t+1}|}{\sum_{t=1}^{T-1} |U_t|}, \tag{1}$$

  where U_t is the set of parameter indices updated at episode t.
- **Cluster Density (CD)**: Evaluates how tightly parameter updates cluster within the model's global parameter index space. Lower CD signifies highly localized updates conducive to Rowhammer effects.

$$\text{CD} = \frac{L}{|\boldsymbol{\theta}|}, \quad \text{where } L = \min_{1 \le j \le k-m+1} (i_{(j+m-1)} - i_{(j)} + 1)$$

  denotes the smallest span containing 90% of updates in each episode.

This metric captures how frequently updates concentrate on a single layer. A higher value indicates that one layer receives a disproportionate number of updates.

3) Bit-Flip Validation: Given the DRAM bandwidth BW, the refresh period $\Delta t_{\text{refresh}}$, and a fraction p that determines the size $S_{\text{update}}(p)$, the maximum number of updates is computed as

$$H_{\text{max}}(p) = \frac{BW \times \Delta t_{\text{refresh}}}{S_{\text{update}}(p)}, \tag{2}$$

where $S_{\text{update}}(p)$ is the size of a sparse update when a fraction p of parameters is transmitted.

To assess the practical feasibility of inducing Rowhammer conditions, we estimate the number of expected activations on a single targeted DRAM row (or closely spaced group of rows) based on empirically measured update clustering behaviors.

The expected number of row activations induced by the repeated updates, denoted as $\mathbb{E}_{Activation}$, can be approximated as:

$$\mathbb{E}_{Activation} = \text{RUR} \times H_{\text{max}}(p), \tag{3}$$

where RUR is the empirically measured repeated update rate

Finally, given a sparse update fraction p, bit-flips are considered successfully triggered when $\mathbb{E}_{Activation}$ is higher than $\overline{T_{\text{flip}}}$.

C. Target System

The target system is a self-supervised federated voice recognition system, to emulate large scale voice assistant pre-training systems, which are widely deployed on various platforms including smart home devices, mobile phones, and other IoT devices, and involving large quantity of clients.

1) Model Structure: We train the agent with target FL ASR systems of the following five model structures, the size details of these models are presented in Table I.

2) Dataset: For dataset, the FL takes the Common Voice 17.0 dataset [39] as the training data, which is a large-scale multilingual voice dataset. We only use the English subset for simplicity. The subset contains 1.1M voice clips with a total duration of more than 3k hours. The data is randomly split into equal client partitions, where the size of the partition is determined by the number of clients.

3) Server Setups: On the server side, global model and aggregation are performed using huge page and page-locked memory, and the communication between the server and clients are optimized using RDMA.

TABLE I
COMMON ASR MODELS WITH THEIR SIZES AND COMPUTATION PRECISION.

Model	Params(M)	Individual Tensors	Precision	$H_{\max}(0.0005)$
Conformer-CTC-S [34]	8.7	480	INT8	296K
QuartzNet 5x5 [35]	6.7	130	INT8	384K
Citrinet 256 [36]	10.3	635	INT8	250K
Squeezeformer-XS [37]	9.0	480	INT8	286K
Eff. Conformer-CTC [38]	13.2	520	INT8	196K

4) Clients Setups: On the client side, each client emulates different physical environments via applying random Gaussian noise in addition to the adversarial noise generated by the RL agent, after which the mixed waveforms are resampled to 16kHz to match the Common Voice dataset. Also, the clients are set to send sparse updates, i.e., at each communication round, only a controllable portion of the most significant gradient updates are sent to the server.

D. RL Agent Design

We propose a two-stage PPO-based agent to generate adversarial waveforms that consistently induce parameter updates clustered within targeted indices of an ASR model's parameter vector, denoted by $\boldsymbol{\theta} \in \mathbb{R}^M$. The two-stage design initially provides explicit guidance for parameter localization (Stage 1), followed by generalized adversarial capability under limited supervision (Stage 2).

*1) **Observation and Action Spaces**:* In **Stage 1**, observations explicitly encode parameter update locations $\mathbf{u}_t$ to facilitate clear mapping from adversarial perturbations to model updates in initial learning, aligning with curriculum-learning principles [40]:

- **Observation**: $\mathcal{O}_1 = \{x_{\text{clean}}, \mathbf{u}_t\}$, with binary vector $\mathbf{u}_t \in \{0,1\}^M$ indicating recently updated parameters.
- **Action**: $\mathcal{A}_1 = \{\delta \in \mathbb{R}^D \mid \|\delta\|_\infty \leq \epsilon\}$, constrained adversarial perturbations.

In **Stage 2**, explicit update signals are removed to prevent dependence on instantaneous signals, promoting robust generalization across diverse inputs:

- **Observation**: $\mathcal{O}_2 = \{x_{\text{clean}}\}$, relying solely on waveform input to generalize learned perturbation strategies.
- **Action**: Maintained as $\mathcal{A}_2 = \mathcal{A}_1$.

*2) **Reward Formulation**:* Both stages share a unified reward structure composed of three components balancing localization, clustering specificity, and imperceptibility constraints:

$$R_t = \underbrace{\alpha\,\text{EMD}(\mathbf{u}_t, \mathbf{u}_{t-1})}_{\text{Update Stability}} + \underbrace{\beta\,\frac{\|\mathbf{u}_t[i:j]\|_0}{\|\mathbf{u}_t\|_0}}_{\text{Target Focus}} - \underbrace{\gamma\mathcal{L}_{\text{perc}}}_{\text{Stealth}} \quad (4)$$

Update Proximity: Earth Mover's Distance (EMD) penalizes large shifts between consecutive update indices, promoting stable parameter clustering.

Target Specificity: Ratio term emphasizes updates within targeted parameter indices $[i:j]$.

Perceptibility Penalty:

$$\mathcal{L}_{\text{perc}} = \lambda_1 \|\text{STFT}(x') - \text{STFT}(x)\|_F + \lambda_2\,\text{rms}(\delta) \quad (5)$$

This perceptual constraint simultaneously limits frequency-domain artifacts and temporal waveform distortions [10].

*3) **Policy Network Architecture**:* The policy network employs a hierarchical architecture with a shared feature extractor and stage-specific decoders. The network first processes raw waveforms through a multi-resolution convolutional frontend, followed by bidirectional temporal modeling, before diverging into stage-dependent perturbation generators.

Shared Encoder Design: The encoder processes raw waveforms through parallel convolutional branches with progressively larger kernels (3, 5, and 25 samples). These branches extract localized phoneme features (4kHz resolution) and broader spectral patterns (1kHz resolution), concatenated into a 1536-channel representation. A 4-layer bidirectional LSTM then models temporal dependencies across 500ms windows, matching common ASR systems.

Stage-Specific Generators:

- *Stage 1* explicitly incorporates parameter update locations by projecting the binary mask $\mathbf{u}_t$ into a 1024-dimensional embedding, concatenated with encoder outputs. This design follows curriculum learning principles, accelerating early-stage association between perturbations and targeted parameter updates.
- *Stage 2* replaces explicit masking with an 8-head cross-attention mechanism. A learned query vector $q \in \mathbb{R}^{512}$ interacts with encoder features through scaled dot-product attention, enabling the agent to focus on temporal regions most correlated with persistent parameter clustering. Both stages employ identical transposed convolutional blocks for perturbation synthesis.

*4) **Training Protocol**:* **Curriculum Phases**: The phased training aligns with the policy network's dual-head architecture to optimize parameter targeting:

1) *Stage 1 (50k episodes)*: This phase trains the shared encoder and Stage 1 head to establish baseline correlations between waveform perturbations and localized parameter updates.
2) *Transition (10k episodes)*: Observation masking $\mathbf{u}_t$ is progressively replaced with zero vectors. The cross-attention head is warm-started using Stage 1 encoder weights, with orthogonal regularization ($\beta = 0.01$) applied to BiLSTM layers to prevent feature collapse.
3) *Stage 2 (20k episodes)*: Full $\mathcal{O}_2$ observations with $\epsilon = 0.1$ enhance attack potency. The attention query vector q is fine-tuned while freezing the shared encoder's first three BiLSTM layers, balancing generalization with temporal feature retention.

TABLE II
ROWHAMMER ATTACK THRESHOLDS UNDER DIFFERENT PATTERNS

Access Pattern	Data Pattern (Victim)	Data Pattern (Aggressor)	Threshold (ACTs)
Single-sided	0x00000000	0xFFFFFFFF	240K
Single-sided	0xFFFFFFFF	0x00000000	185K
Single-sided	0x55555555	0x55555555	260K
Single-sided	0xAAAAAAAA	0xAAAAAAAA	265K
Double-sided	0x00000000	0xFFFFFFFF	140K
Double-sided	0xFFFFFFFF	0x00000000	115K
Double-sided	0x55555555	0x55555555	160K
Double-sided	0xAAAAAAAA	0xAAAAAAAA	165K

IV. EXPERIMENTS

A. Experimental Setups

This work utilises the DRAM Bender open-source platform to test the Rowhammer attack threshold [41]. The platform is built on the Alveo U200 Accelerated Card [42] and is designed to emulate a memory controller using an FPGA, allowing us to precisely issue Rowhammer-related instructions. The specific memory module tested is the MTA18ASF2G72PZ-2G3B1-16GB [43], which complies with the standard DDR4 specification. Additionally, it has data rate of 2400 MT/s and bit-width of 72 bits. After excluding 8 bits of ECC, the effective bit-width for transmission is 64 bits:

$$BW = \frac{\text{Data Rate} \times \text{Bit-Width}}{8 \text{ bits/byte}} \tag{6}$$

According to equation (6), we can first get the bandwidth of tested module is 18.75 GB/s. Therefore, the data throughput for each 64ms refresh period will be approximately 1.2 GB. Furthermore, the default **tRC** is 46.16 ns.

In an operating system-level attack environment, the optimal `ACT` rate based on the `clflushopt` instruction sequence is 159 ACTs/tREFI, meaning that the actual average **tRC** used for frequent activations is 49 ns, which is close to the theoretical ACT limit. Under this condition, the attacker can issue up to 1,306K activations per refresh window. Moreover, all other timing parameters of tested module follow the default configuration.

B. Rowhammer Test Bench

Considering that FL may use different memory replacement strategies and that data diversity plays a role, this work tested different Rowhammer attack thresholds, based on two access patterns and four data patterns. Compare to single-sided access pattern, the double-sided requires less activations. However, it is also more difficult to utilise as it requires two adjacent rows for victim data. The data pattern will be the data layout between victim rows and aggressor rows. The tests not only reveal patterns that are more likely to induce bit flips, such as $0 \rightarrow 1$ or $1 \rightarrow 0$, but also demonstrate relatively more challenging patterns, such as $0 \rightarrow 0$ or $1 \rightarrow 1$. Specially, we use hexadecimal strings to represent data patterns that fills the entire DRAM row. For example, `0xFFFFFFFF` represents 32-bit data filled with all ones, while `0x00000000` represents all zeros. The obtained attack thresholds are derived from the average values across all banks of the tested DRAM module.

These activation thresholds ensure that the attack induces bit-flips in at least 95% of the DRAM rows. The inability to obtain thresholds for all rows is likely due to the presence of exceptionally resilient cells within the module, which results in extreme deviations in the average threshold. These thresholds are specifically presented in Table II. All threshold values fall within the theoretical limit of 1,309K activations and reserve enough dummy rows activations to bypass TRR. It can be observed that double-sided Rowhammer generally requires fewer activations than single-sided Rowhammer. In the worst case, 265K activations are sufficient to induce bit flips.

C. PPO Agent Training

We empirically evaluate our proposed PPO-based adversarial framework to assess its effectiveness in consistently inducing clustered and repetitive parameter updates in federated Automatic Speech Recognition (ASR) systems under varying sparsity settings, which are suitable for Rowhammer exploitation.

1) Experimental Setup: We selected five widely-used ASR architectures of varying complexity, parameter counts, and architectural patterns for evaluation, see in Table I the five models and their parameter statistics.

For each model, two levels of parameter-update sparsity were considered:

- **Medium Sparsity (0.1% updates):** Represents scenarios where moderate communication efficiency is required without severely limiting model convergence speed.
- **High Sparsity (0.05% updates):** Reflects extreme communication efficiency conditions, simulating highly bandwidth-constrained or large-scale deployments.

The PPO agent was trained to generate adversarial waveforms constrained by a maximum perturbation limit ($\|\delta\|_\infty \leq 0.1$), ensuring the adversarial audio samples remain stealthy and practically imperceptible. The training dataset employed was the English subset of Common Voice 17.0 [39]. Hyperparameters for the reward function were empirically optimized as $\alpha = 1.0$, $\beta = 0.8$, and $\gamma = 0.6$, with perceptibility penalty weights set to $\lambda_1 = \lambda_2 = 0.5$.

2) Baseline Methods: To provide clear insights into the effectiveness of our two-stage PPO approach, we compared it against two baselines:

- **Single-Stage PPO (PPO-SS):** PPO trained skipping stage 1 for the same number of total episodes.
- **Unperturbed**: FL operating without any external influences.

3) Evaluation Metrics: To evaluate our PPO agent, we measure their performance with the following metrics, previously defined in III-D.

- Repeated Update Rate (RUR)
- Cluster Density (CD)
- Average Rowhammer threshold across all rows ($\overline{T_{\text{flip}}}$)

4) Results and Analysis: Tables III and IV present comprehensive evaluations of our PPO-TS method against baseline approaches across different model architectures and sparsity levels.

TABLE III
REPEATED TENSOR OVERLAP RATE (RUR, %) ACROSS ALL MODELS AND SPARSITY LEVELS.

Model	Method	RUR (0.1%)	RUR (0.05%)
Conformer-CTC-S	PPO-TS	73.7	68.3
	PPO-SS	65.1	53.9
	Unperturbed	12.5	9.3
QuartzNet 5x5	PPO-TS	75.6	71.1
	PPO-SS	64.8	57.2
	Unperturbed	11.8	8.7
Citrinet 256	PPO-TS	71.9	66.4
	PPO-SS	57.4	52.9
	Unperturbed	13.9	10.2
Squeezeformer-XS	PPO-TS	72.5	68.1
	PPO-SS	61.4	54.4
	Unperturbed	11.2	8.1
Eff. Conformer-CTC	PPO-TS	71.1	66.1
	PPO-SS	59.3	51.7
	Unperturbed	13.2	9.6

Repeated Update Rate (RUR): As summarized in Table III, our PPO-TS approach consistently outperformed both the PPO-SS and Unperturbed baselines. At high sparsity (0.05%), PPO-TS achieved RUR ranging from 66.1% (Eff. Conformer-CTC) to 71.1% (QuartzNet 5x5), significantly surpassing PPO-SS (approximately 51% – 57%). This substantial improvement underscores the two stage design to learn robust, long-term strategies for generating perturbations that repeatedly affect the same parameters.

Cluster Density (CD): As detailed in Table IV, PPO-TS resulted in significantly lower CD values compared to baselines across all tested configurations. At the high sparsity level, PPO-TS achieved CD values as low as 1.0%–1.6%, marking approximately a 2–3× reduction compared to PPO-SS (around 3.4%–4.1%). Such extreme clustering effectively concentrates updates into tightly bound memory regions, dramatically increasing the probability of repeatedly activating the same DRAM rows.

These results demonstrate the efficacy of our approach in systematically inducing concentrated and stable update patterns necessary for facilitating targeted Rowhammer attacks in realistic FL settings.

D. Bit-Flip Validation

In this subsection, we jointly analyze the effectiveness of our proposed PPO-based adversarial attack framework in inducing practical Rowhammer conditions, considering both the theoretical Rowhammer thresholds obtained in Section IV-B and the empirically measured clustering results from PPO agent training presented in Section IV-C. Specifically, we quantify the practical feasibility of triggering Rowhammer bit-flips by correlating the observed Repeated Update Rate (RUR) with the number of sparse gradient updates performed in federated learning scenarios.

TABLE IV
CLUSTER DENSITY (CD, %) ACROSS ALL MODELS AND SPARSITY LEVELS.

Model	Method	CD (%)	
		0.1%	0.05%
Conformer-CTC-S	PPO-TS	1.2	0.6
	PPO-SS	3.7	1.8
	Unperturbed	19.2	15.4
QuartzNet 5x5	PPO-TS	1.5	0.8
	PPO-SS	4.1	3.2
	Unperturbed	18.6	14.8
Citrinet 256	PPO-TS	1.0	0.7
	PPO-SS	3.4	1.9
	Unperturbed	17.9	13.9
Squeezeformer-XS	PPO-TS	1.6	0.9
	PPO-SS	3.8	2.4
	Unperturbed	19.5	15.8
Eff. Conformer-CTC	PPO-TS	1.1	0.5
	PPO-SS	3.6	2.3
	Unperturbed	18.2	14.3

This straightforward estimation facilitates practical reasoning about the potential for successful Rowhammer attacks based on experimentally observed adversarial clustering.

1) Estimated Row Activation Counts: Table V summarizes the calculated expected row activations for all evaluated ASR models at the medium sparsity setting (0.1%), comparing our two-stage PPO method (**PPO-TS**) against single-stage PPO (**PPO-SS**) and Unperturbed baseline methods.

2) Estimated Bit Flips: Refer to Table II, averaging bit flip thresholds over different single sided aggressor patterns, bit flips are expected when repeated activation exceeds 240K, with more vulnerable cells flips at 200K, i.e., $\overline{T_{\text{flip}}} = 240,000$. Our agent's attack on QuartzNet 5x5 with sparsity of 0.05% can be deemed successful in inducing Rowhammer bit flips, and there is a high possibility of success in inducing Rowhammer bit flips through attacking Conformer-CTC-S.

V. DISCUSSION

In this section, we discuss the practical implications, limitations, security considerations, and mitigation strategies related to our proposed PPO-driven Rowhammer attack against federated learning (FL) systems.

A. Practical Implications and Feasibility

Our experiments provide empirical evidence that PPO-generated physical-domain perturbations can effectively induce clustered and repetitive parameter updates critical for successful Rowhammer exploitation in federated learning scenarios. Particularly at higher sparsity levels in large-scale FL

TABLE V
ESTIMATED ROW ACTIVATIONS BASED ON EMPIRICAL CLUSTERING (TOP 0.05% SPARSITY LEVEL)

Model	Method	$H_{\max}(0.0005)$	RUR (%)	$\mathbb{E}_{Activations}$
Conformer-CTC-S	PPO-TS	296,000	68.3	202,168
	PPO-SS		53.9	159,544
	Unperturbed		9.3	27,528
QuartzNet 5x5	PPO-TS	384,000	71.1	273,024
	PPO-SS		57.2	219,648
	Unperturbed		8.7	33,408
Citrinet 256	PPO-TS	250,000	66.4	166,000
	PPO-SS		52.9	132,250
	Unperturbed		10.2	25,500
Squeezeformer-XS	PPO-TS	286,000	68.1	194,766
	PPO-SS		54.4	155,584
	Unperturbed		8.1	23,166
Eff. Conformer-CTC	PPO-TS	196,000	66.1	129,556
	PPO-SS		56.9	111,524
	Unperturbed		9.6	18,816

deployments, our PPO-TS approach significantly outperforms baseline methods by achieving high repetition rates of over 70% and highly localized cluster densities. Such performance substantially increases the likelihood of repeatedly activating specific DRAM rows, thus enhancing the practical feasibility of successful Rowhammer bit-flips.

However, practical real-world deployment may introduce complexities such as partial system knowledge, environmental variability, or limited precision in physical perturbation generation. Despite these uncertainties, our joint evaluation clearly demonstrates that the activation counts achieved by our PPO-based attacks are realistically within the threshold ranges required to trigger Rowhammer attacks, particularly under favorable DRAM configurations or minor system optimizations (e.g., higher memory bandwidth or extended refresh intervals). Therefore, our results underscore a significant and realistic security risk to federated learning systems utilizing common performance optimization practices.

B. Limitations and Open Challenges

We acknowledge two primary limitations affecting the practical applicability of the proposed PPO-driven Rowhammer attack:

1) **Opacity of Target FL Setups:** Real-world federated systems typically obscure internal details such as aggregation policies, memory management, and system optimizations from external attackers. Without explicit feedback or partial system knowledge, the current reinforcement learning framework might experience significantly reduced efficiency and effectiveness. Future research could address this limitation by developing RL agents with highly sparse rewards or inferable rewards, or transferable adversarial perturbation techniques.
2) **Lack of Control over Specific Victim Rows:** The proposed indirect Rowhammer attack inherently lacks precise control over the specific DRAM rows affected by induced bit-flips, making outcomes unpredictable and potentially transient. This uncertainty reduces the reliability of achieving specific attack objectives (e.g., targeted data corruption or privilege escalation). Future research should explore advanced side-channel techniques or hardware-software co-design approaches to improve the precision and predictability of the Rowhammer effects induced by physical-domain perturbations.

C. Security Considerations for Federated Learning Systems

Our findings highlight significant security risks arising from commonly used performance optimization techniques in federated learning systems, including sparse updates, RDMA, huge pages, and pinned memory. While these optimizations greatly enhance computational and communication efficiency, they also inadvertently create predictable and sustained DRAM activation patterns exploitable by hardware-level adversaries. Consequently, FL practitioners must carefully reconsider optimization-related trade-offs, explicitly evaluating hardware security vulnerabilities when designing and deploying federated learning systems, especially in sensitive or critical infrastructure scenarios.

D. Potential Mitigation Strategies

Based on our comprehensive analysis and findings, we propose several practical defensive measures to mitigate the identified threat:

- **Adversarial Input Defense:** From the sofware prospective, as a variant of adversarial attack, defense measures, such as input space transformation, feature denoising and adversarial input detection and rejection, are also effective. The actual effectiveness of these measures depends on the tolerance of delay on clients end, as well as adapting settings to the variant of adversarial attack. Input space transformation and feature denoising can filter adversarial perturbations of lower magnitudes, while perturbations of higher magnitudes are relatively easier to detect and reject.

- **Hardware-Level Countermeasures:** From the hardware perspective, upgrading to DRAM modules with enhanced Rowhammer resilience, such as those with advanced error correction, higher refresh frequency, offers direct defense against Rowhammer exploitation attempts of all sources.
- **Systematic Analysis of Optimization Technique Deployment:** When deploying new optimization techniques, systematically assess and analyze the effects of new techniques in combination of existing optimization techniques with considerations of both software execution and hardware characteristics, hence avoiding inherent security risks and lowering the feasibility of the proposed attack vector.

E. Broader Impact and Future Directions

Our study demonstrates a compelling need for interdisciplinary approaches combining hardware security, adversarial machine learning, and federated learning systems design. By integrating hardware-level vulnerabilities into FL threat modeling, we advocate a holistic view that proactively identifies, assesses, and mitigates emerging threats.

Future research directions include:

- Extending our approach to diverse model architectures, and real-world applications through stronger agents that can process side channel information and learn with highly sparse rewards in real-world applications to enhance the attack.
- Exploring hardware-software joint-design approaches that mitigate Rowhammer risks without severely compromising computational efficiency.

By addressing these open challenges and continuing to bridge security gaps between hardware vulnerabilities and distributed learning systems, we aim to contribute toward more secure, dependable, and resilient federated learning deployments.

VI. Conclusion

In this paper, we introduced and comprehensively evaluated a novel Rowhammer attack vector targeting federated learning (FL) systems, exploiting physically driven interference orchestrated by a Proximal Policy Optimization (PPO)-based reinforcement learning agent. Unlike traditional Rowhammer attacks, our method leverages adversarial acoustic or electromagnetic perturbations at client sensors, indirectly inducing clustered and repetitive parameter updates on FL servers. Our empirical results demonstrate the efficacy of our approach, particularly at higher sparsity levels common in realistic FL deployments, effectively increasing the frequency of repeated DRAM row activations critical for triggering Rowhammer bitflips.

Through extensive evaluation and analysis across diverse Automatic Speech Recognition (ASR) architectures, we demonstrated that the proposed two-stage PPO approach substantially enhances adversarial capabilities compared to baseline methods, achieving both higher repeated update rates and more concentrated update clustering. Our joint evaluation highlighted practical feasibility, with achieved activation counts realistically approaching Rowhammer thresholds under favorable conditions, emphasizing the tangible security threat posed by this attack vector.

Furthermore, we analyzed the security implications associated with common FL optimizations such as sparse updates, pinned memory, and RDMA, illustrating how performance-oriented design choices can unintentionally expose systems to hardware-level vulnerabilities. We outlined practical mitigation strategies, including randomized memory allocation, adversarial input detection, and controlled usage of optimization techniques, as proactive defensive measures to counteract this emerging threat.

Our findings underscore the necessity for interdisciplinary threat modeling, encompassing both software and hardware domains, to ensure the secure and dependable deployment of federated learning systems.

References

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*. PMLR, 2017, pp. 1273–1282.

[2] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings *et al.*, "Advances and open problems in federated learning," *Foundations and trends® in machine learning*, vol. 14, no. 1–2, pp. 1–210, 2021.

[3] Y. Kim, R. Daly, J. Kim, C. Fallin, J. H. Lee, D. Lee, C. Wilkerson, K. Lai, and O. Mutlu, "Flipping bits in memory without accessing them: An experimental study of dram disturbance errors," *ACM SIGARCH Computer Architecture News*, vol. 42, no. 3, pp. 361–372, 2014.

[4] P. Frigo, E. Vannacc, H. Hassan, V. Van Der Veen, O. Mutlu, C. Giuffrida, H. Bos, and K. Razavi, "Trrespass: Exploiting the many sides of target row refresh," in *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2020, pp. 747–762.

[5] P. Jattke, M. Wipfli, F. Solt, M. Marazzi, M. Bölcskei, and K. Razavi, "Zenhammer: Rowhammer attacks on amd zen-based platforms," in *33rd USENIX Security Symposium (USENIX Security 2024)*, 2024.

[6] M. Seaborn and T. Dullien, "Exploiting the dram rowhammer bug to gain kernel privileges," *Black Hat*, vol. 15, no. 71, p. 2, 2015.

[7] D. Gruss, C. Maurice, and S. Mangard, "Rowhammer. js: A remote software-induced fault attack in javascript," in *Detection of Intrusions and Malware, and Vulnerability Assessment: 13th International Conference, DIMVA 2016, San Sebastián, Spain, July 7-8, 2016, Proceedings 13*. Springer, 2016, pp. 300–321.

[8] A. Tatar, R. K. Konoth, E. Athanasopoulos, C. Giuffrida, H. Bos, and K. Razavi, "Throwhammer: Rowhammer attacks over the network and defenses," in *2018 USENIX Annual Technical Conference (USENIX ATC 18)*, 2018, pp. 213–226.

[9] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno, and D. Song, "Robust physical-world attacks on deep learning visual classification," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 1625–1634.

[10] N. Carlini and D. Wagner, "Audio adversarial examples: Targeted attacks on speech-to-text," in *2018 IEEE security and privacy workshops (SPW)*. IEEE, 2018, pp. 1–7.

[11] O. Mutlu and J. S. Kim, "Rowhammer: A retrospective," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 8, pp. 1555–1571, 2019.

[12] H. Hassan, Y. C. Tugrul, J. S. Kim, V. Van der Veen, K. Razavi, and O. Mutlu, "Uncovering in-dram rowhammer protection mechanisms: A new methodology, custom rowhammer patterns, and implications," in *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture*, 2021, pp. 1198–1213.

[13] F. de Ridder, P. Frigo, E. Vannacci, H. Bos, C. Giuffrida, and K. Razavi, "{SMASH}: Synchronized many-sided rowhammer attacks from {JavaScript}," in *30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 1001–1018.

[14] Y. Jiang, H. Zhu, H. Shan, X. Guo, X. Zhang, and Y. Jin, "Trrscope: Understanding target row refresh mechanism for modern ddr protection," in *2021 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. IEEE, 2021, pp. 239–247.

[15] Z. B. Aweke, S. F. Yitbarek, R. Qiao, R. Das, M. Hicks, Y. Oren, and T. Austin, "Anvil: Software-based protection against next-generation rowhammer attacks," *ACM SIGPLAN Notices*, vol. 51, no. 4, pp. 743–755, 2016.

[16] L. Cojocar, K. Razavi, C. Giuffrida, and H. Bos, "Exploiting correcting codes: On the effectiveness of ecc memory against rowhammer attacks," in *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2019, pp. 55–71.

[17] L. Cojocar, J. Kim, M. Patel, L. Tsai, S. Saroiu, A. Wolman, and O. Mutlu, "Are we susceptible to rowhammer? an end-to-end methodology for cloud providers," in *2020 IEEE symposium on security and privacy (SP)*. IEEE, 2020, pp. 712–728.

[18] Y. Xiao, X. Zhang, Y. Zhang, and R. Teodorescu, "One bit flips, one cloud flops:{Cross-VM} row hammer attacks and privilege escalation," in *25th USENIX security symposium (USENIX Security 16)*, 2016, pp. 19–35.

[19] I. Kang, W. Wang, J. Kim, S. van Schaik, Y. Tobah, D. Genkin, A. Kwong, and Y. Yarom, "{SledgeHammer}: Amplifying rowhammer via bank-level parallelism," in *33rd USENIX Security Symposium (USENIX Security 24)*, 2024, pp. 1597–1614.

[20] P. Pessl, D. Gruss, C. Maurice, M. Schwarz, and S. Mangard, "{DRAMA}: Exploiting {DRAM} addressing for {Cross-CPU} attacks," in *25th USENIX security symposium (USENIX security 16)*, 2016, pp. 565–581.

[21] R. Qiao and M. Seaborn, "A new approach for rowhammer attacks," in *2016 IEEE international symposium on hardware oriented security and trust (HOST)*. IEEE, 2016, pp. 161–166.

[22] Y. Jang, J. Lee, S. Lee, and T. Kim, "Sgx-bomb: Locking down the processor via rowhammer attack," in *Proceedings of the 2nd Workshop on System Software for Trusted Execution*, 2017, pp. 1–6.

[23] M. Lipp, M. Schwarz, L. Raab, L. Lamster, M. T. Aga, C. Maurice, and D. Gruss, "Nethammer: Inducing rowhammer faults through network requests," in *2020 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*. IEEE, 2020, pp. 710–719.

[24] C. Szegedy, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[25] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[26] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," in *Artificial intelligence safety and security*. Chapman and Hall/CRC, 2018, pp. 99–112.

[27] T. B. Brown, D. Mané, A. Roy, M. Abadi, and J. Gilmer, "Adversarial patch," *arXiv preprint arXiv:1712.09665*, 2017.

[28] I. Evtimov, K. Eykholt, E. Fernandes, T. Kohno, B. Li, A. Prakash, A. Rahmati, and D. Song, "Robust physical-world attacks on machine learning models," *arXiv preprint arXiv:1707.08945*, vol. 2, no. 3, p. 4, 2017.

[29] M. Sharif, S. Bhagavatula, L. Bauer, and M. K. Reiter, "Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition," in *Proceedings of the 2016 acm sigsac conference on computer and communications security*, 2016, pp. 1528–1540.

[30] A. Athalye, L. Engstrom, A. Ilyas, and K. Kwok, "Synthesizing robust adversarial examples," in *International conference on machine learning*. PMLR, 2018, pp. 284–293.

[31] R. Duan, X. Ma, Y. Wang, J. Bailey, A. K. Qin, and Y. Yang, "Adversarial camouflage: Hiding physical-world attacks with natural styles," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 1000–1008.

[32] X. Yuan, Y. Chen, Y. Zhao, Y. Long, X. Liu, K. Chen, S. Zhang, H. Huang, X. Wang, and C. A. Gunter, "{CommanderSong}: a systematic approach for practical adversarial voice recognition," in *27th USENIX security symposium (USENIX security 18)*, 2018, pp. 49–64.

[33] M. Zhou, Z. Qin, X. Lin, S. Hu, Q. Wang, and K. Ren, "Hidden voice commands: Attacks and defenses on the vcs of autonomous driving cars," *IEEE Wireless Communications*, vol. 26, no. 5, pp. 128–133, 2019.

[34] A. Gulati, J. Qin, C.-C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu *et al.*, "Conformer: Convolution-augmented transformer for speech recognition," *arXiv preprint arXiv:2005.08100*, 2020.

[35] S. Kriman, S. Beliaev, B. Ginsburg, J. Huang, O. Kuchaiev, V. Lavrukhin, R. Leary, J. Li, and Y. Zhang, "Quartznet: Deep automatic speech recognition with 1d time-channel separable convolutions," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 6124–6128.

[36] S. Majumdar, J. Balam, O. Hrinchuk, V. Lavrukhin, V. Noroozi, and B. Ginsburg, "Citrinet: Closing the gap between non-autoregressive and autoregressive end-to-end models for automatic speech recognition," *arXiv preprint arXiv:2104.01721*, 2021.

[37] S. Kim, A. Gholami, A. Shaw, N. Lee, K. Mangalam, J. Malik, M. W. Mahoney, and K. Keutzer, "Squeezeformer: An efficient transformer for automatic speech recognition," *Advances in Neural Information Processing Systems*, vol. 35, pp. 9361–9373, 2022.

[38] M. Burchi and V. Vielzeuf, "Efficient conformer: Progressive downsampling and grouped attention for automatic speech recognition," in *2021 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2021, pp. 8–15.

[39] R. Ardila, M. Branson, K. Davis, M. Henretty, M. Kohler, J. Meyer, R. Morais, L. Saunders, F. M. Tyers, and G. Weber, "Common voice: A massively-multilingual speech corpus," in *Proceedings of the 12th Conference on Language Resources and Evaluation (LREC 2020)*, 2020, pp. 4211–4215.

[40] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proceedings of the 26th Annual International Conference on Machine Learning*, ser. ICML '09. New York, NY, USA: Association for Computing Machinery, 2009, p. 41–48. [Online]. Available: https://doi.org/10.1145/1553374.1553380

[41] A. Olgun, H. Hassan, A. G. Yağlıkçı, Y. C. Tuğrul, L. Orosa, H. Luo, M. Patel, O. Ergin, and O. Mutlu, "Dram bender: An extensible and versatile fpga-based infrastructure to easily test state-of-the-art dram chips," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 42, no. 12, pp. 5098–5112, 2023.

[42] "Alveo u200 and u250 data center accelerator cards data sheet (ds962)," accessed: 28-Feb-2025. [Online]. Available: https://docs.amd.com/r/en-US/ds962-u200-u250

[43] "Mta18asf2g72pz-2g3b1 datasheet," accessed: 28-Feb-2025. [Online]. Available: https://www.mouser.co.uk/datasheet/2/671/asf18c2gx72pz-3079314.pdf