DeepSteal: Advanced Model Extractions Leveraging Efficient Weight Stealing in Memories

Adnan Siraj Rakin*[1], Md Hafizul Islam Chowdhuryy*[2], Fan Yao[2], Deliang Fan[1]

*Joint First Authors

[1]Arizona State University (asrakin@asu.edu, dfan@asu.edu)

Website: https://dfan.engineering.asu.edu/

[2]University of Central Florida (reyad@knights.ucf.edu, Fan.Yao@ucf.edu)

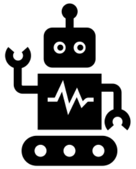
Website: https://casrl.ece.ucf.edu

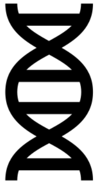
Outline

- Background
- Threat Model and Overview
- System-level Attack
- Substitute Model Training
- Experimental Setup
- Results & Conclusion

Machine Learning (ML) Applications

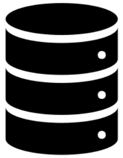
- Machine Learning Applications:

• Robotics 

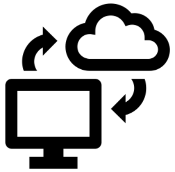
• Medical Applications 

• Self-Driving Cars 

- Machine Learning Cloud Services:

• Amazon AWS AI 

• Google AI 

• Microsoft Azure ML 

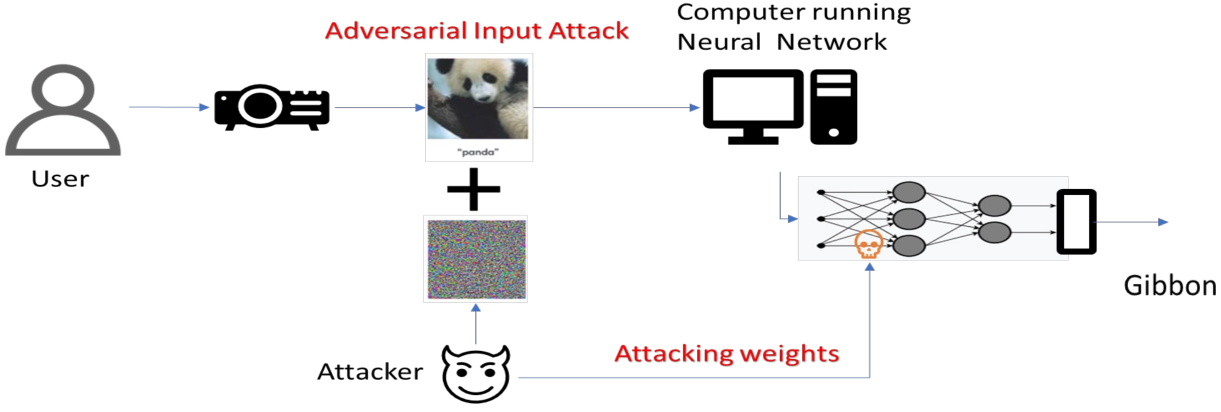
Adversarial Threats in ML:

Model Tampering (Security)

- External Threat (Input perturbation)
  Adversarial Examples (Madry et. al. ICLR-18)

- Internal Threat (Weight perturbation)
  DeepHammer (Yao et. al. USENIX SEC-20)

- Both (Trojan/Backdoor Attack)
  Trojan NN (Liu et. al. NDSS-18)



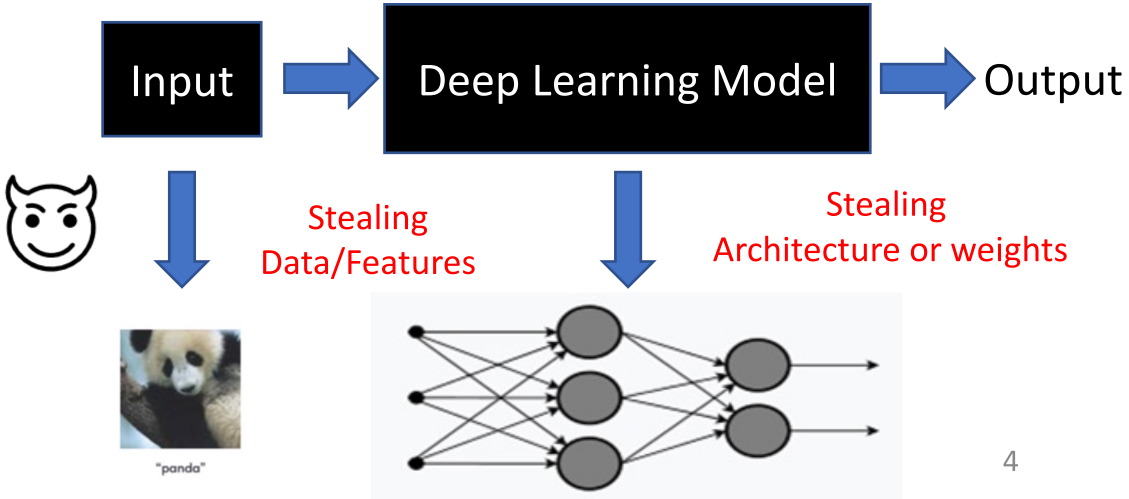
[Rakin et. al. USENIX SEC-21]

Model Leakage (Privacy)

- Model Inversion Attack
  Recover Data (Fredrikson et. al. CCS-15)

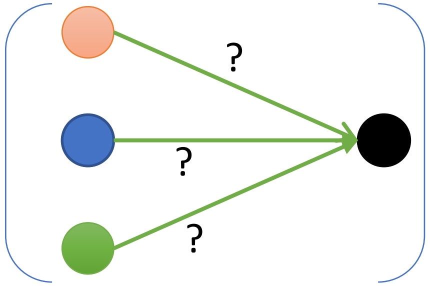
- Membership Inference Attack
  Leak Training Data (Shokri et. al. S&P-17)

- **Model Extraction Attack *Recover Model Architecture/ Weights***
  DeepSniffer (Hu et. al. ASPLOS-20)

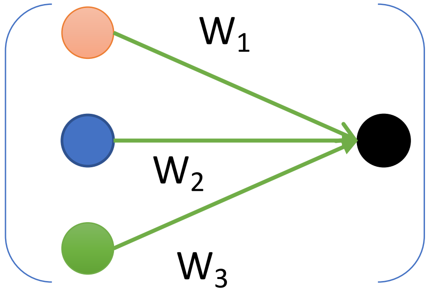


Model Extraction Attack Objective:

Victim's Target
DNN Model



Goal of our DeepSteal:
Recover W_1,W_2,W_3



1. Create a substitute model to ***mimic the functionality*** of the targe model with ***limited dataset*** (less than 10 %).

2. The substitute model should have a high ***accuracy and fidelity***.

3. The substitute model can generate ***strong transferable adversarial examples*** to attack the target model.

Remote Side channel Attack on ML Model

Primary Goal of Prior Works:

Recover model architecture (i.e., no. of layers/connections)

Example:

Cache telepathy [USENIX Security'20], ***DeepSniffer*** [ASPLOS'20]

Opportunities:

1. None of the existing remote side-channel works have successfully recovered fine-grained weight information.

2. Exfiltration of weight information can potentially be even more dangerous than leakage of architecture information.

Can we recover fine-grained weight information through the remote side channels?

How to utilize partial weight information to perform advanced model extraction?

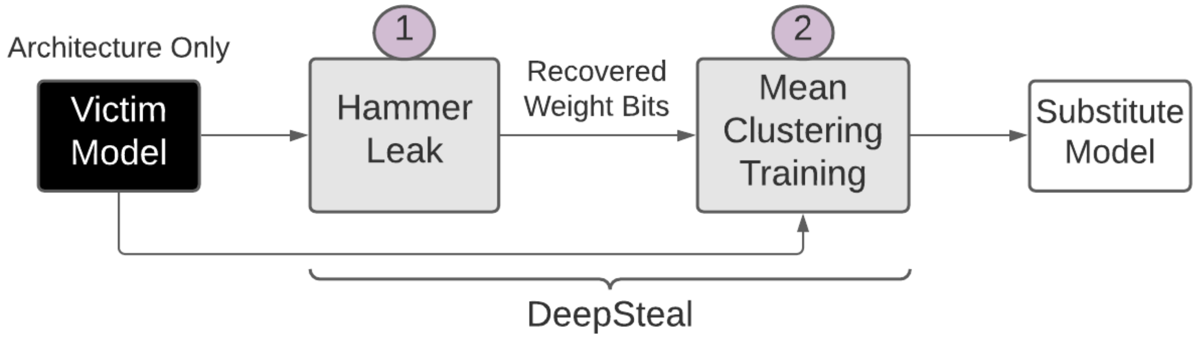
Outline

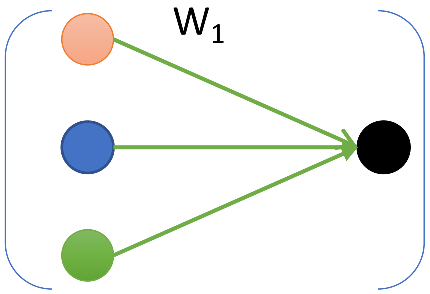
- Background
- Threat Model and Overview
- System-level Attack
- Substitute Model Training
- Experimental Setup
- Results & Conclusion

Threat Model

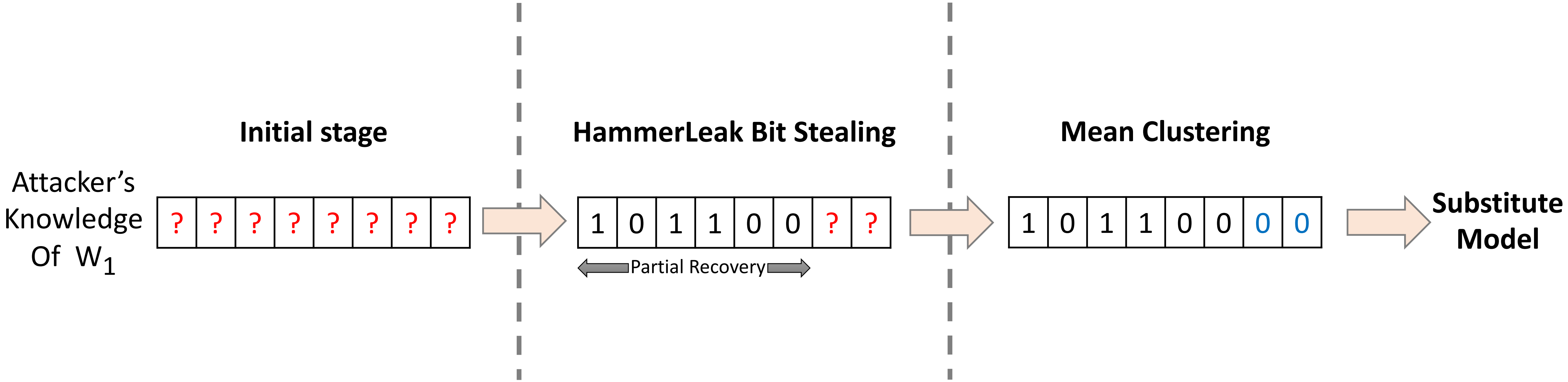
- Attacker *knows* the DNN model architecture.
- Attacker *does not know* gradient or model parameter information.
- Attacker *cannot query* the target model to get output scores.
- Attacker can run *userspace process* on the victim machine.
- System software are *benign and properly protected*.

DeepSteal Overview





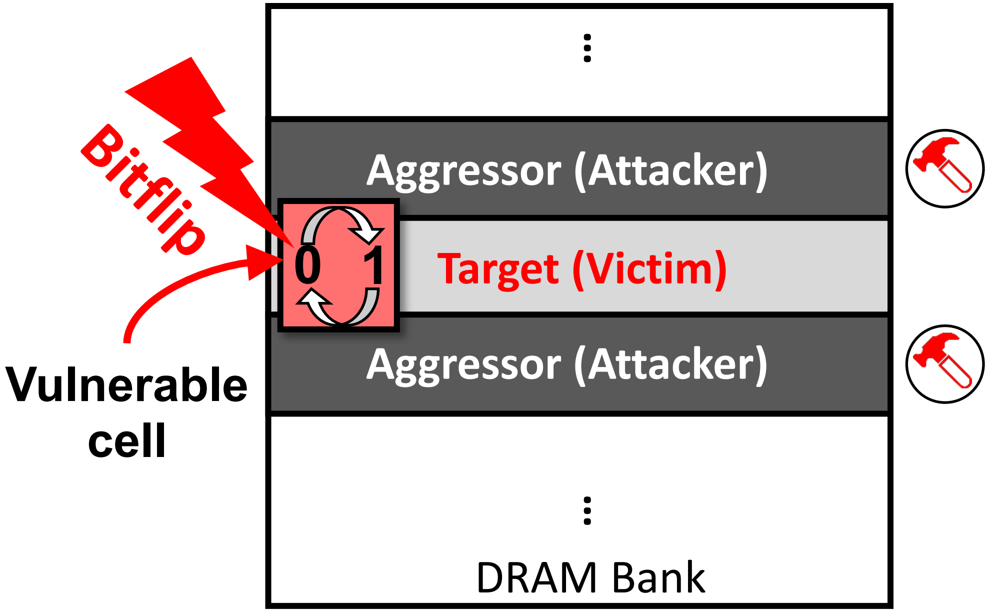
Victim's target model



Outline

- Background
- Threat Model and Overview
- System-level Attack
- Substitute Model Training
- Experimental Setup
- Results & Conclusion

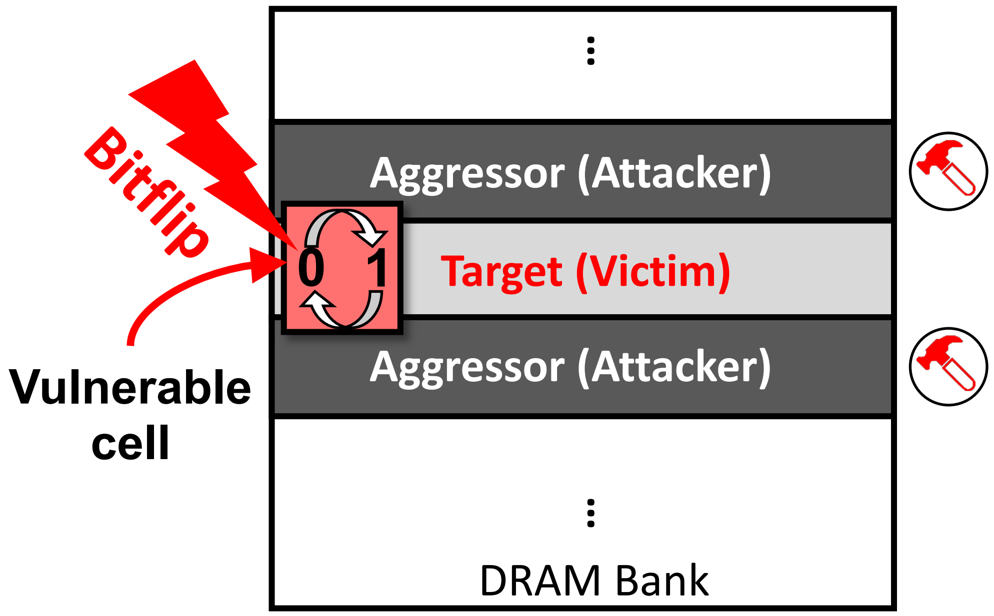
Data Leakage through RowHammer



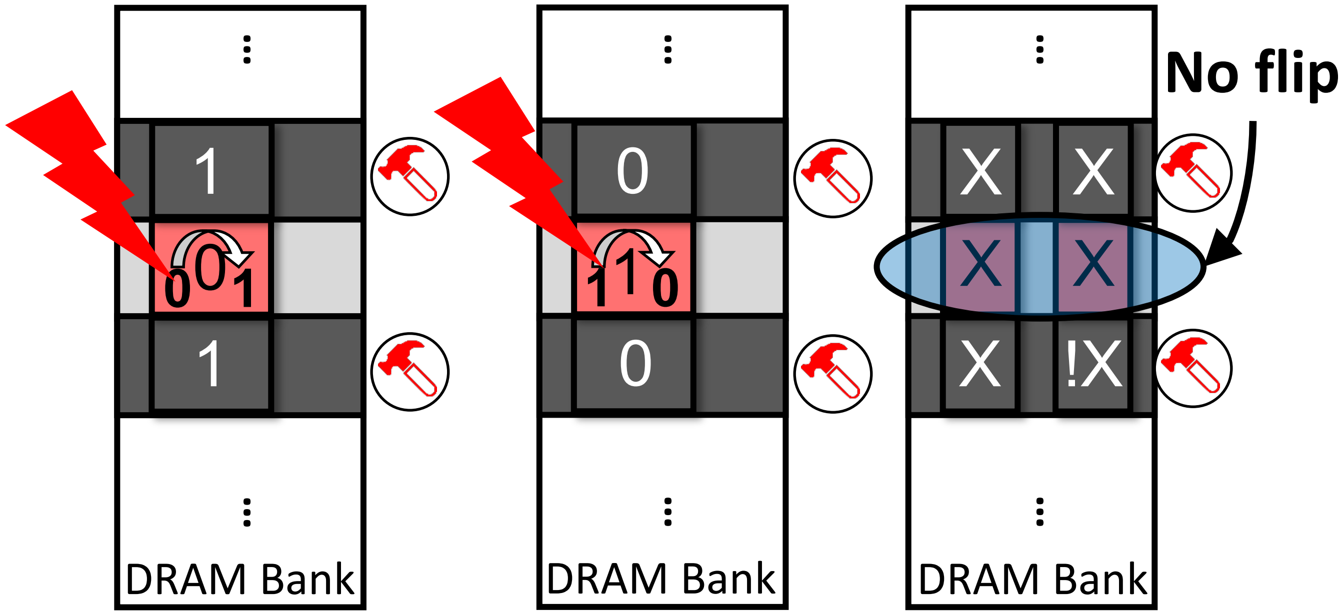
RowHammer-based
fault injection

Bitflips are ***data dependent***

Data Leakage through RowHammer

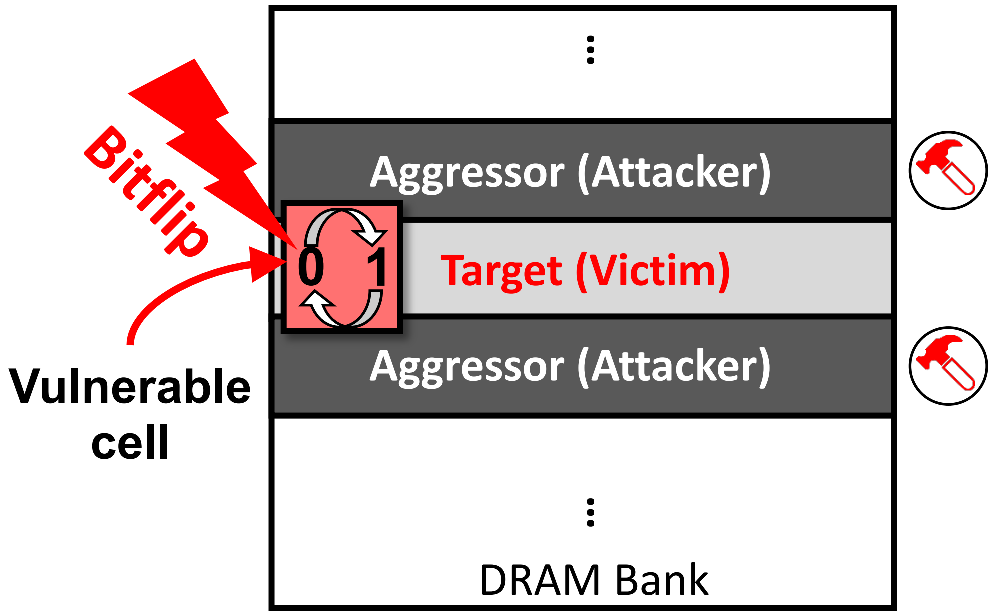


RowHammer-based
fault injection

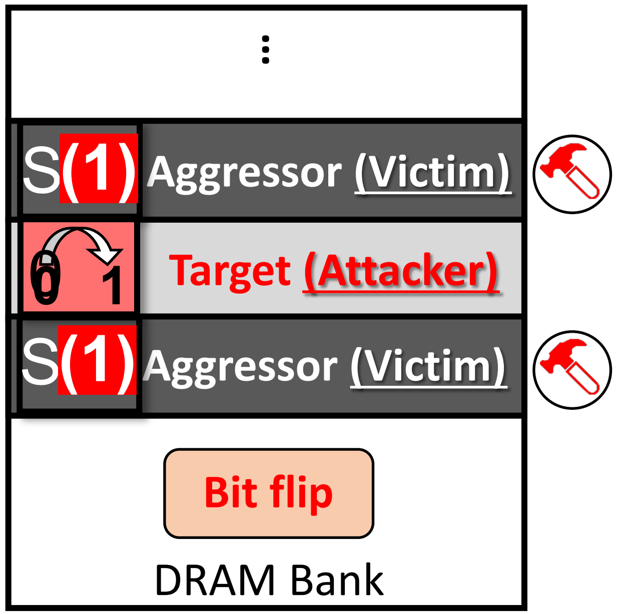


Bitflips are ***data dependent***

Data Leakage through RowHammer



RowHammer-based
fault injection



RowHammer-based **information leakage** (S -> Secret)

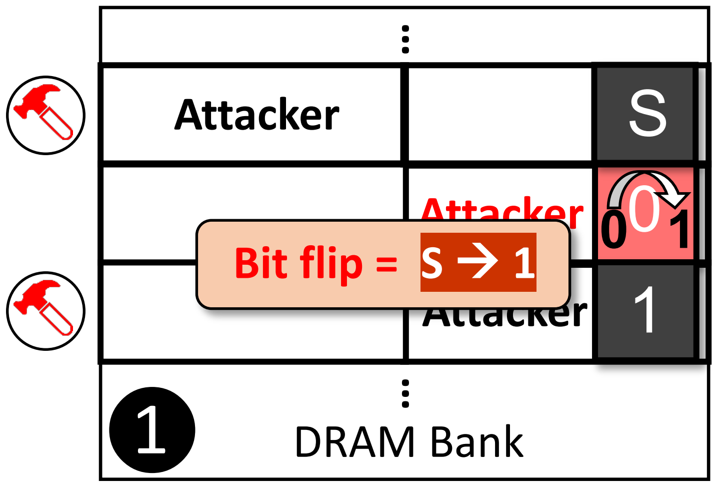
Aggressor bit can be leaked based on the existence of bitflip (***RAMBleed***, *S&P'20*)

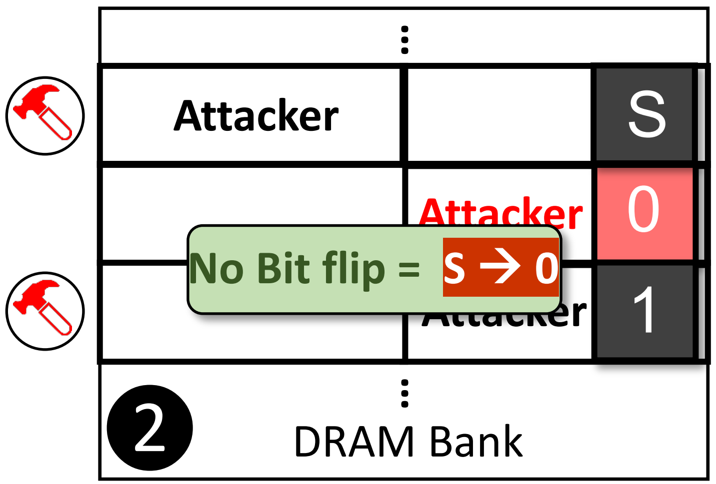
Challenges:

C1: RowHammer information leakage from generic victim application.

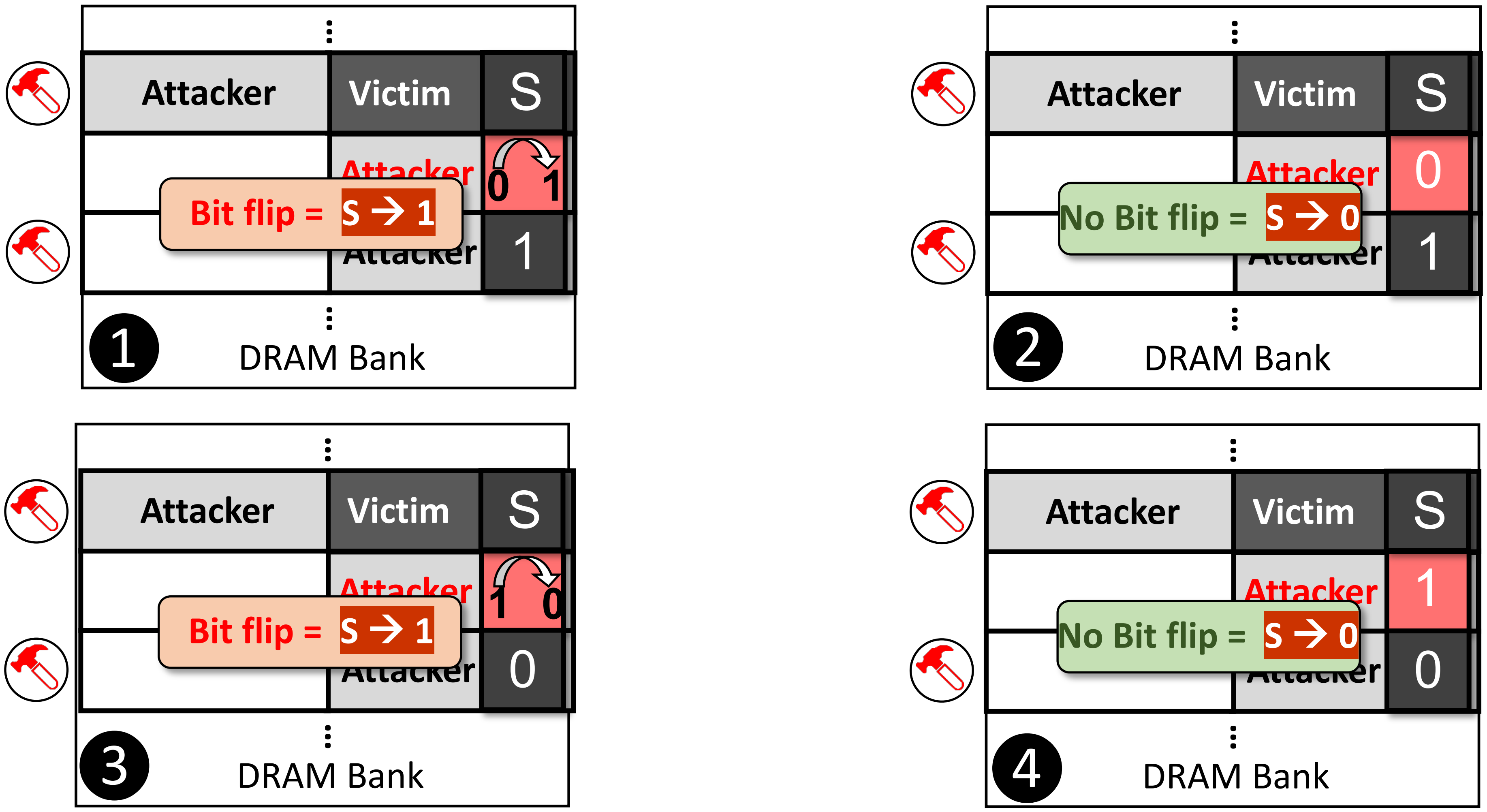
C2: Bulk data stealing from victim with large-scale memory footprint.

Generic RowHammering For Bit Leakage



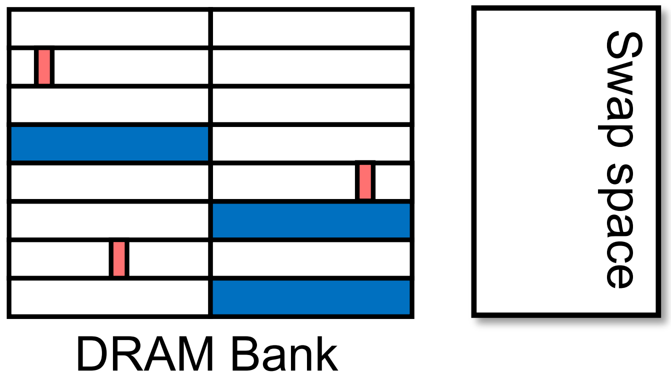


Generic RowHammering For Bit Leakage



HammerLeak Framework: Detailed

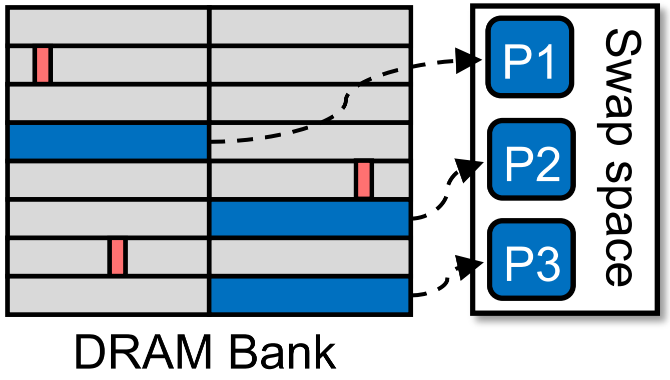
Anonymous
Page Swapping



Free Page Victim Page Attacker Page Vulnerable cell

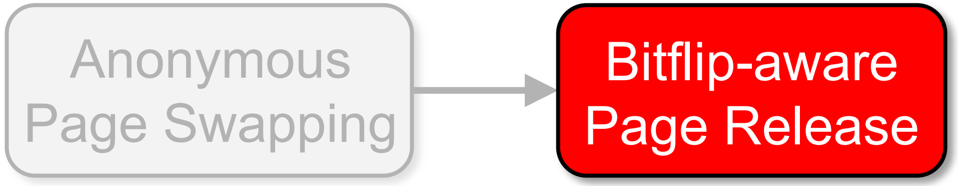
HammerLeak Framework: Detailed

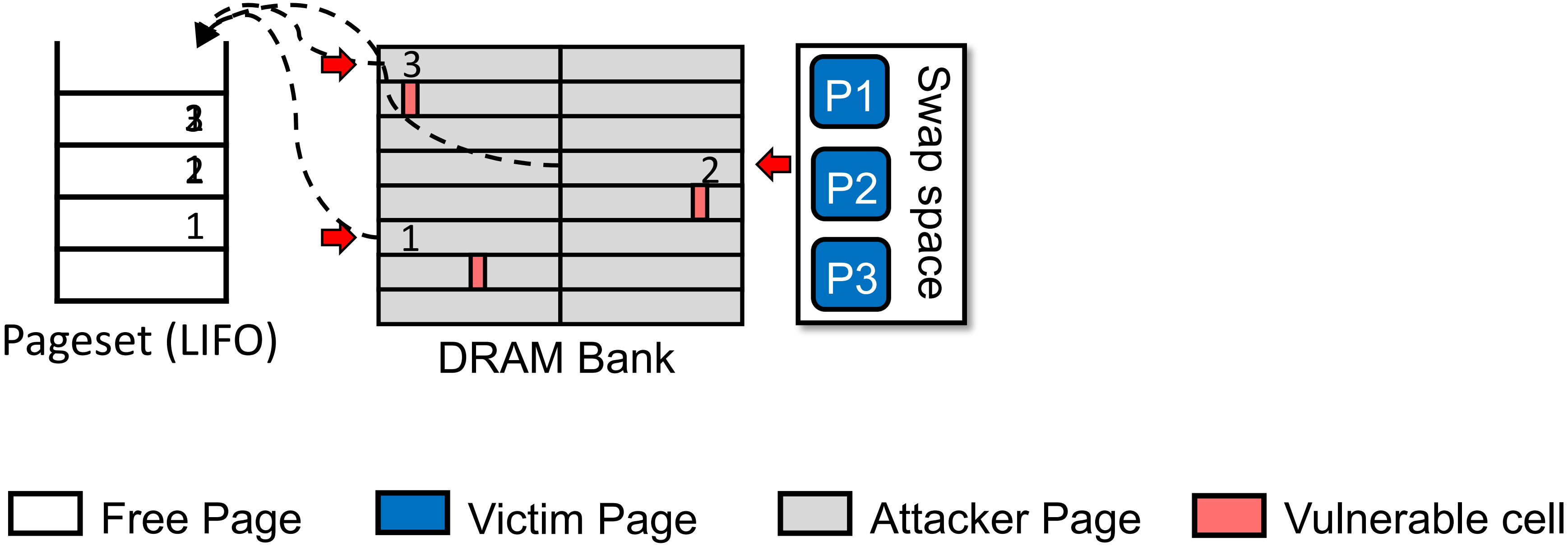
Anonymous Page Swapping



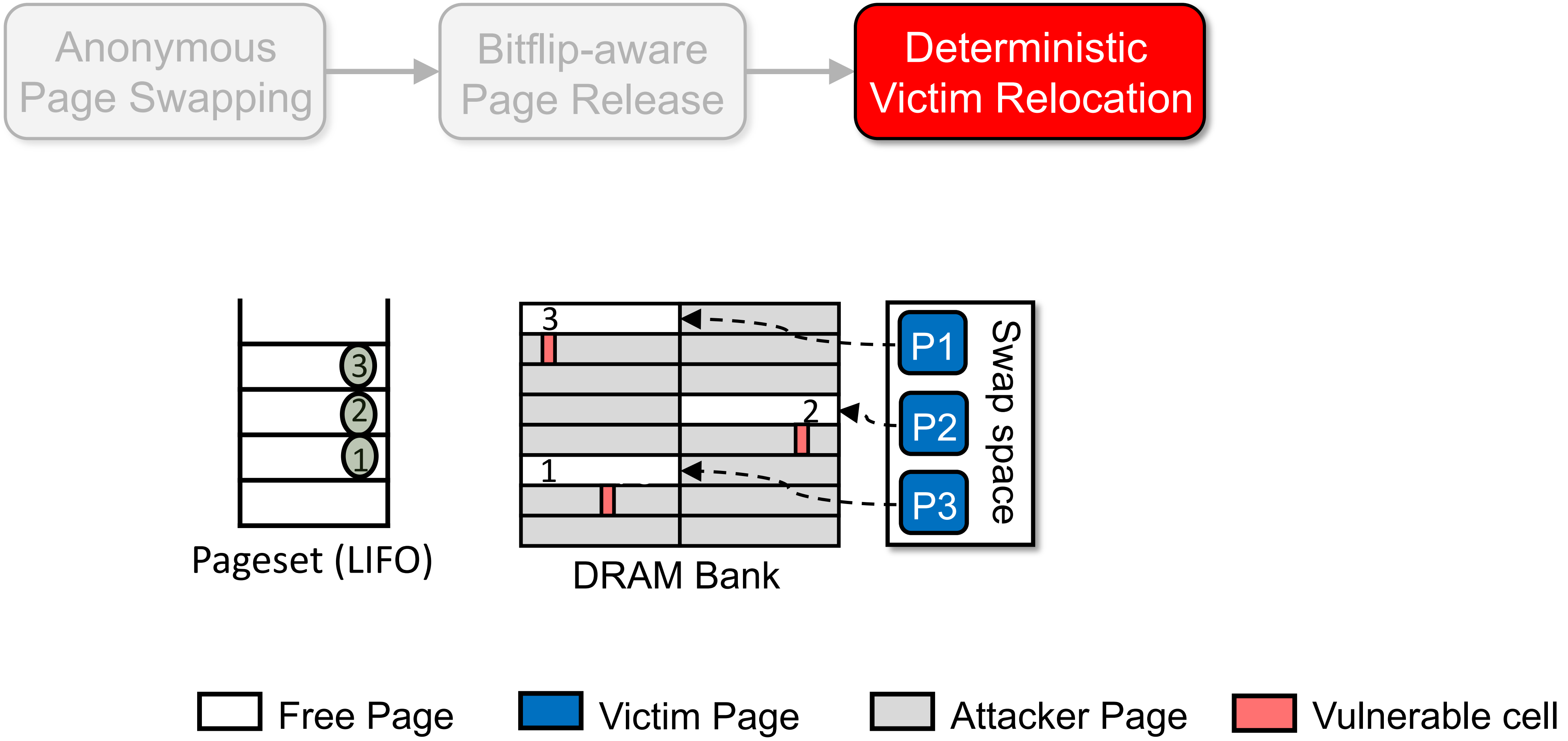
Free Page Victim Page Attacker Page Vulnerable cell

HammerLeak Framework: Detailed



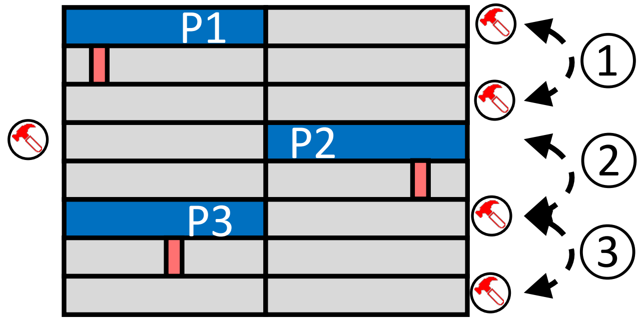


HammerLeak Framework: Detailed



HammerLeak Framework: Detailed

Anonymous Page Swapping → Bitflip-aware Page Release → Deterministic Victim Relocation → **Rowhammer-based Bit Recovery**

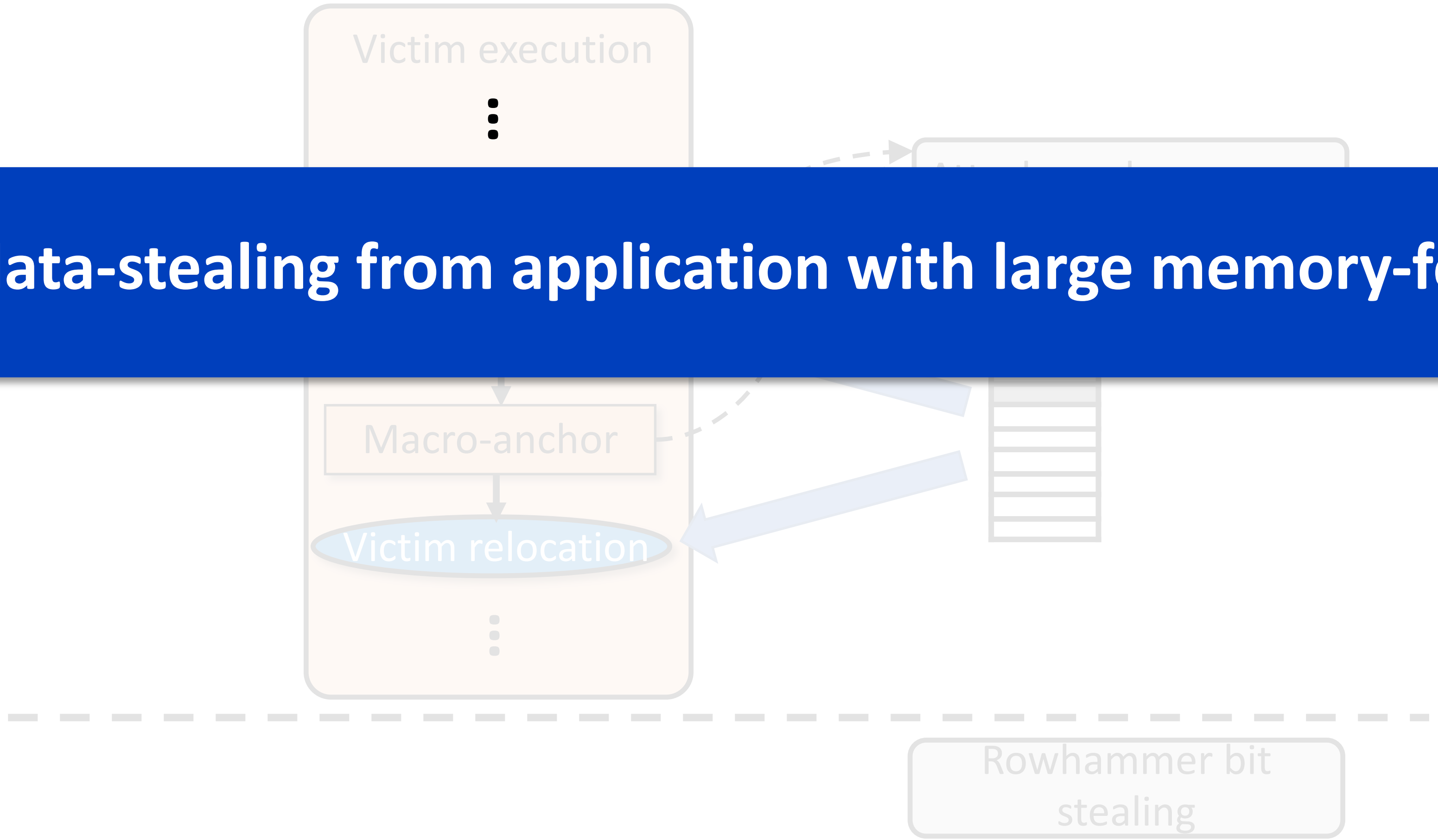


DRAM Bank

Free Page | Victim Page | Attacker Page | Vulnerable cell

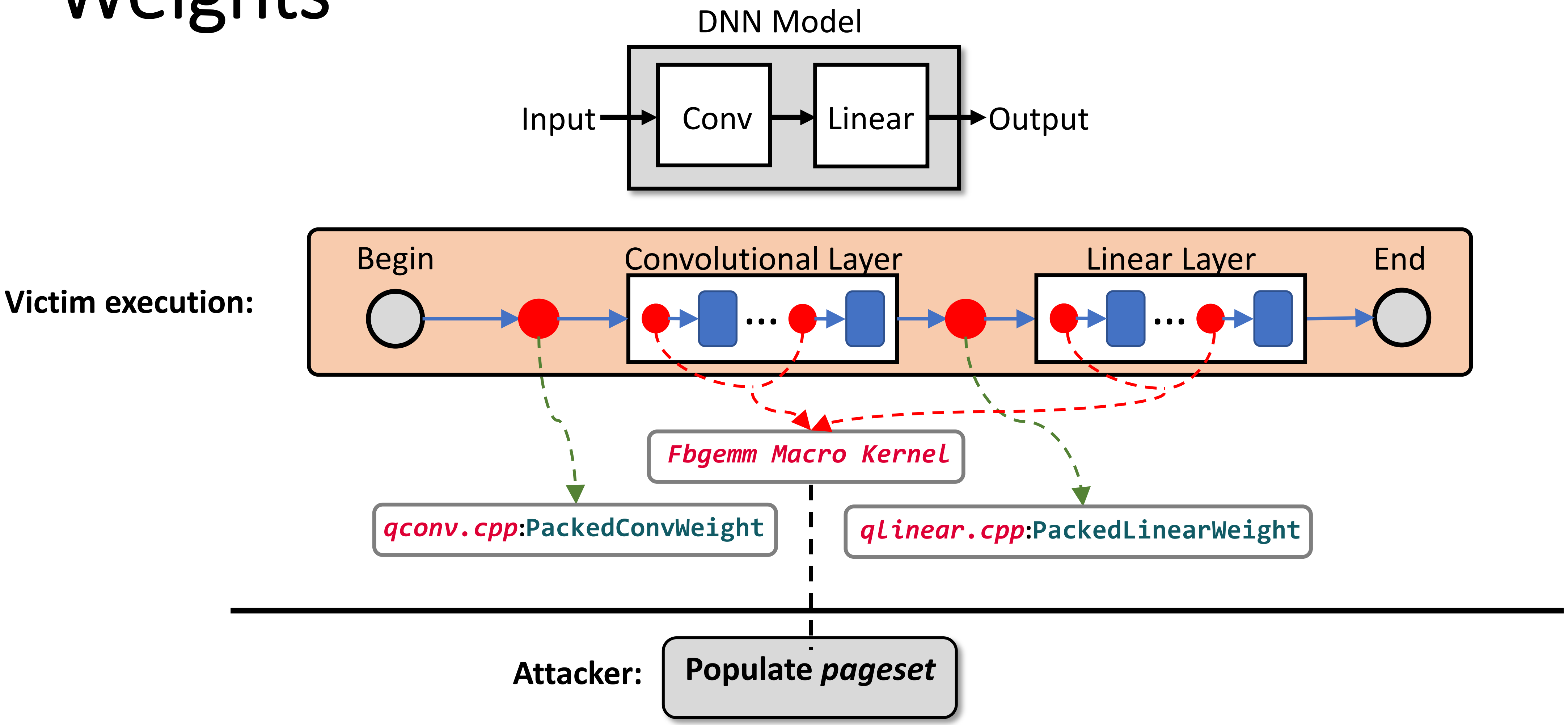
HammerLeak: Batched Page Release

Use smaller batch size: *macro-anchor* to further divide victim execution



Bulk data-stealing from application with large memory-footprint

HammerLeak: Leaking PyTorch Model Weights



Outline

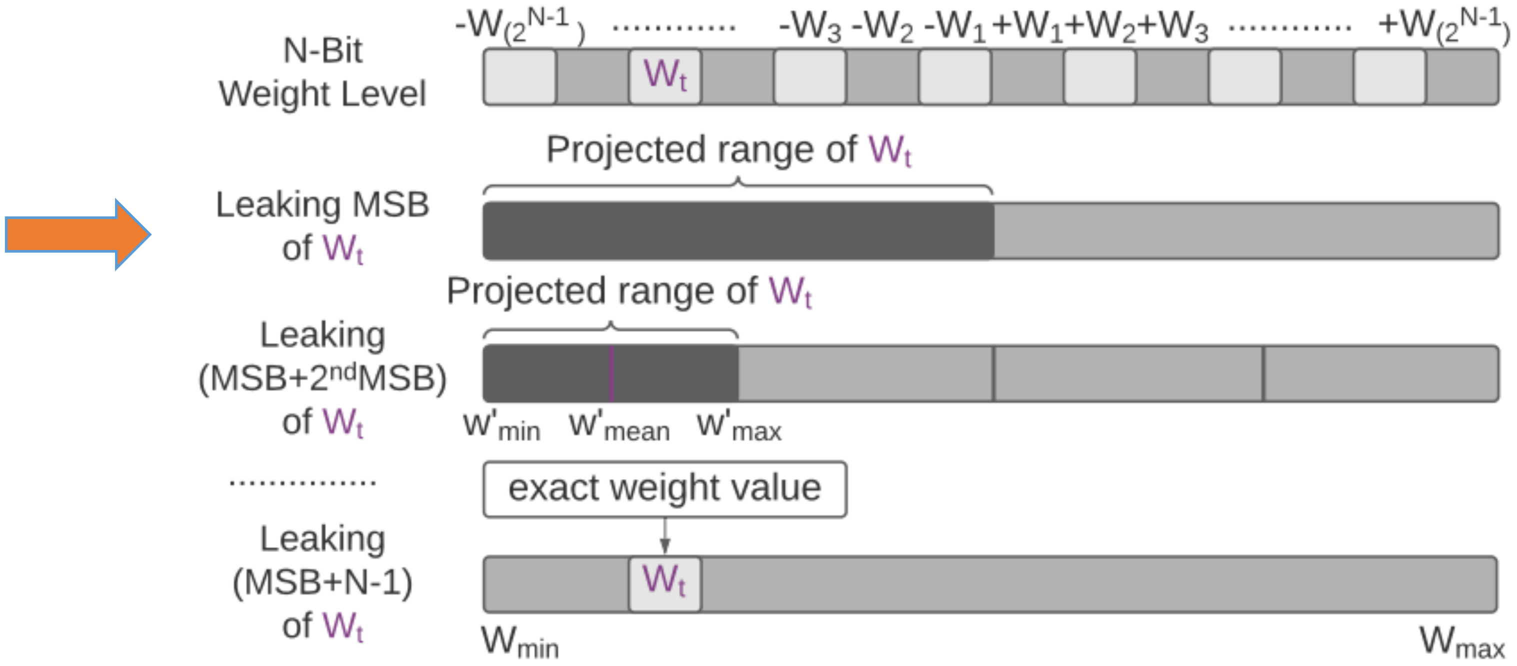
- Background
- Threat Model and Overview
- System-level Attack
- Substitute Model Training
- Experimental Setup
- Results & Conclusion

Why do we still need training?

- ❑ Problem: How to use the partial bit information recovered from HammerLeak?
- ❑ Solution: We propose a training algorithm to successfully utilize the stolen partial bit information.

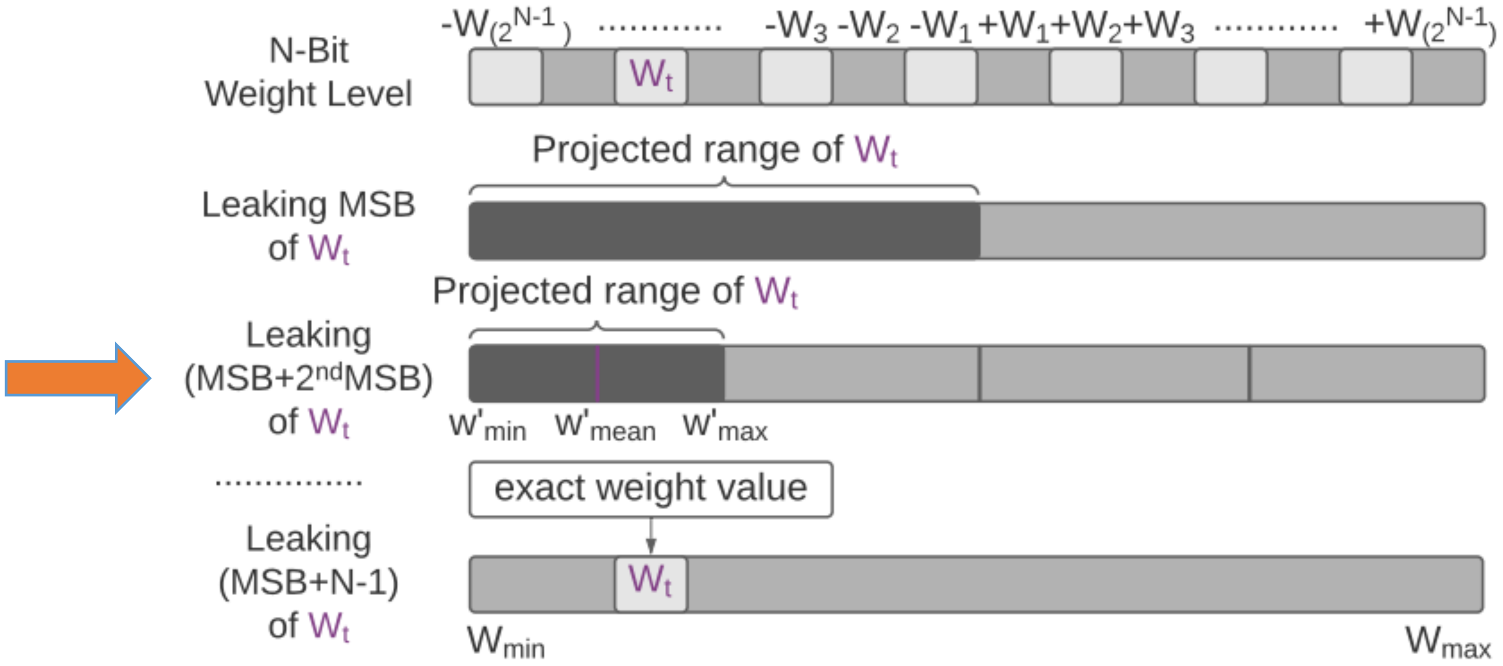
Substitute Model Training:

1. Each weight has a projected range.



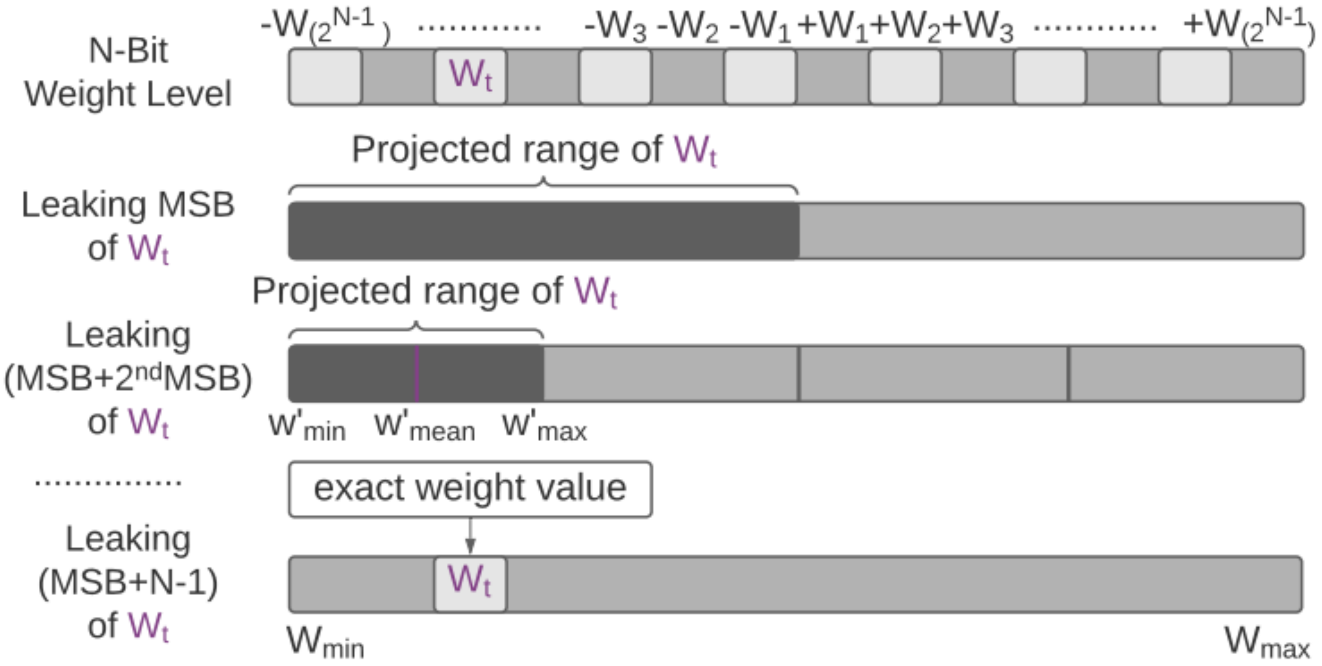
Substitute Model Training:

1. Each weight has a projected range.



Substitute Model Training:

1. Each weight has a projected range.

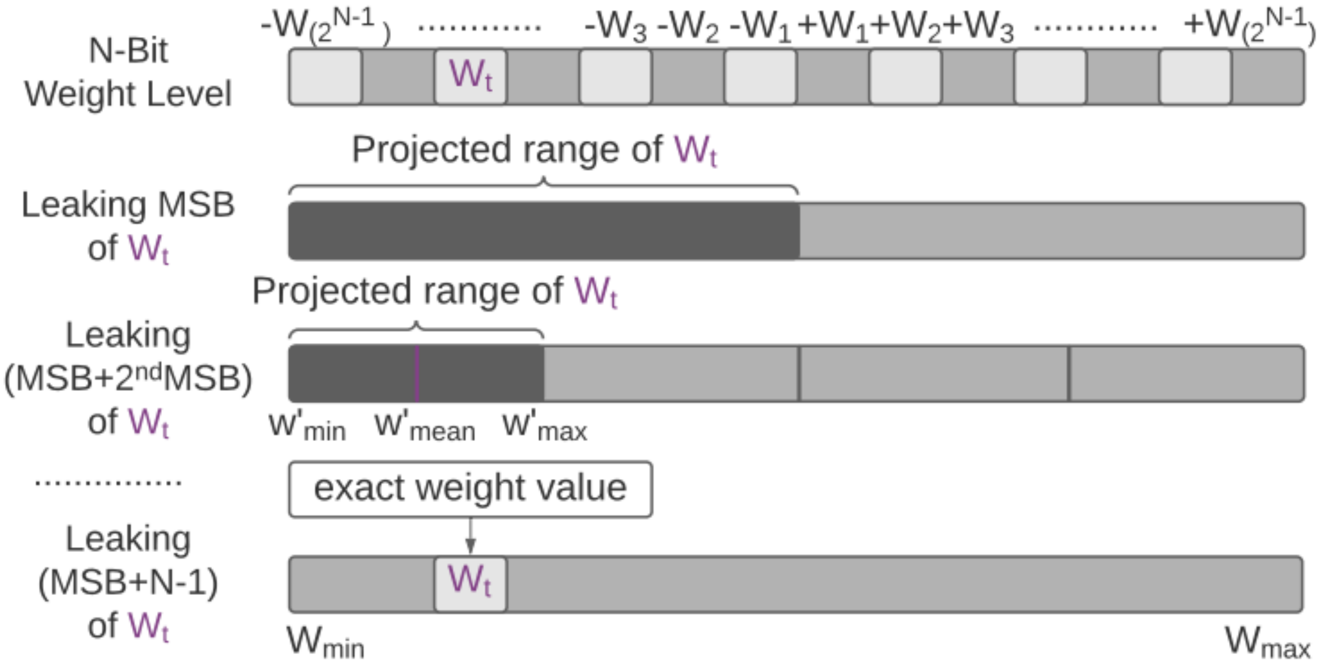


Substitute Model Training:

1. Each weight has a projected range.

2. Mean clustering penalty ensures the weights stay well within the projected range during training.

$$\min_{\{\mathbf{W}_l\}_{l=1}^{L}} \mathbb{E}_{\boldsymbol{x}}\, \mathcal{L}(f(\boldsymbol{x}, \{\mathbf{W}^l\}_{l=1}^{L}), \boldsymbol{y}) + \underbrace{\lambda \cdot \sum_{l=1}^{L} (||\mathbf{W}^l - \mathbf{W}^l_{mean}||)}_{\text{loss penalty for Mean Clustering}}$$



Algorithm: Mean Clustering Training

- Weight Set-1: **All 8-bits recovered**
  No Training i.e., set the gradient of the weights to zero.

- Weight Set-2: **Partial bits recovered starting from most significant bits**
  Apply mean clustering penalty only for these set of weights.

- Weight Set-3: **No bit recovered or bit recovered without MSBs**
  Train w/o any clustering penalty.

Outline

- Background
- Threat Model and Overview
- System-level Attack
- Substitute Model Training
- Experimental Setup
- Results & Conclusion

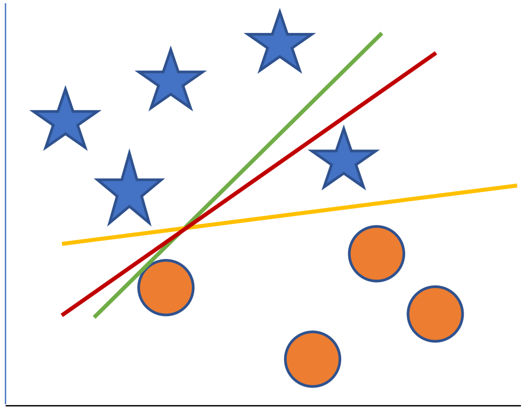
Experimental Setup

- Dataset: Popular vision datasets (e.g., *CIFAR-10/100, GTSRB*).
- Architecture: *ResNets and VGG.*
- Attacker Data: ***8 %*** training data available to train the substitute model.
- Training Platform: *PyTorch* running on *GeForce GTX 1080 Ti GPU* platform.
- Attack Platform: *Intel Haswell* series processor.
- Memory configuration: *Dual-channel DDR3.*

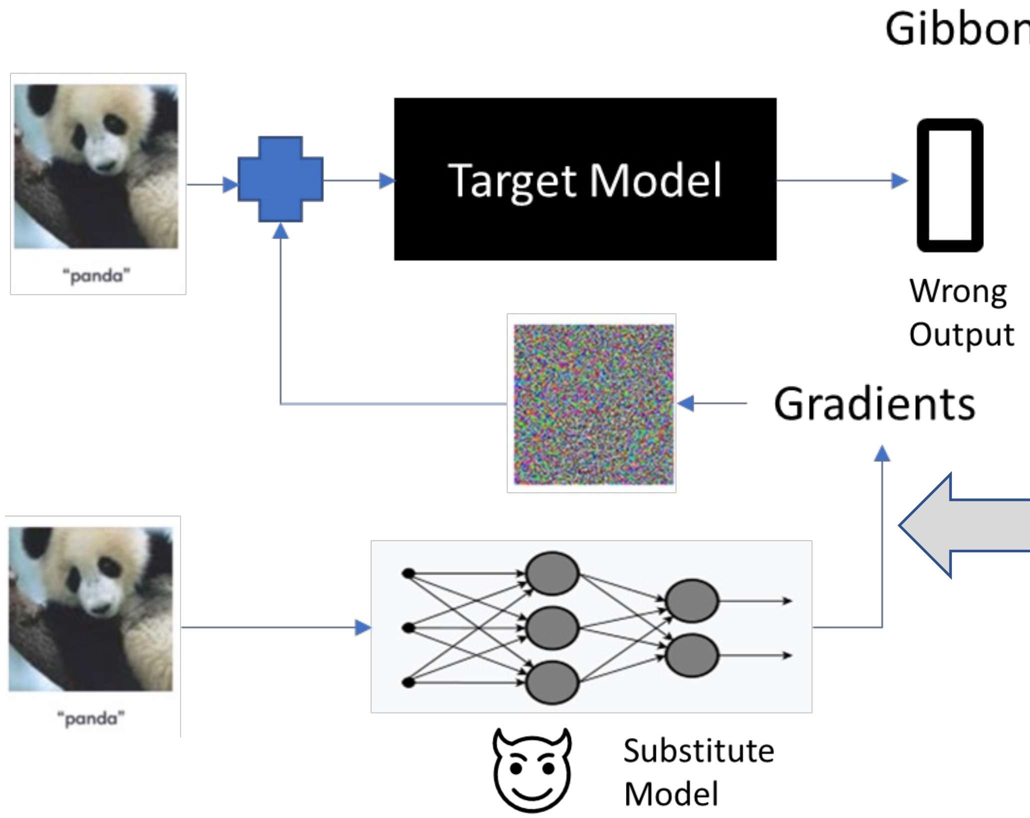
Evaluation Metrics:

Accuracy (%) : Accuracy of the substitute model on test dataset.

Fidelity (%) : Percentage of test samples both the target and substitute model agree on their classification result.



Green: Target Model Decision Boundary
Red: Substitute Model with High Fidelity
Yellow: Substitute Model with High Accuracy



Adversarial Example Attack (%): Test accuracy of a target model on the adversarial test samples generated using the recovered substitute model as shown in the left figure.

Outline

- Background
- Threat Model and Overview
- System-level Attack
- Substitute Model Training
- Experimental Setup
- Results & Conclusion

Results: HammerLeak

HammerLeak Analysis:

- **Bit leakage accuracy:** 95.73% (**Standard deviation:** 0.74%).
- ResNet-18 weight leakage rate.

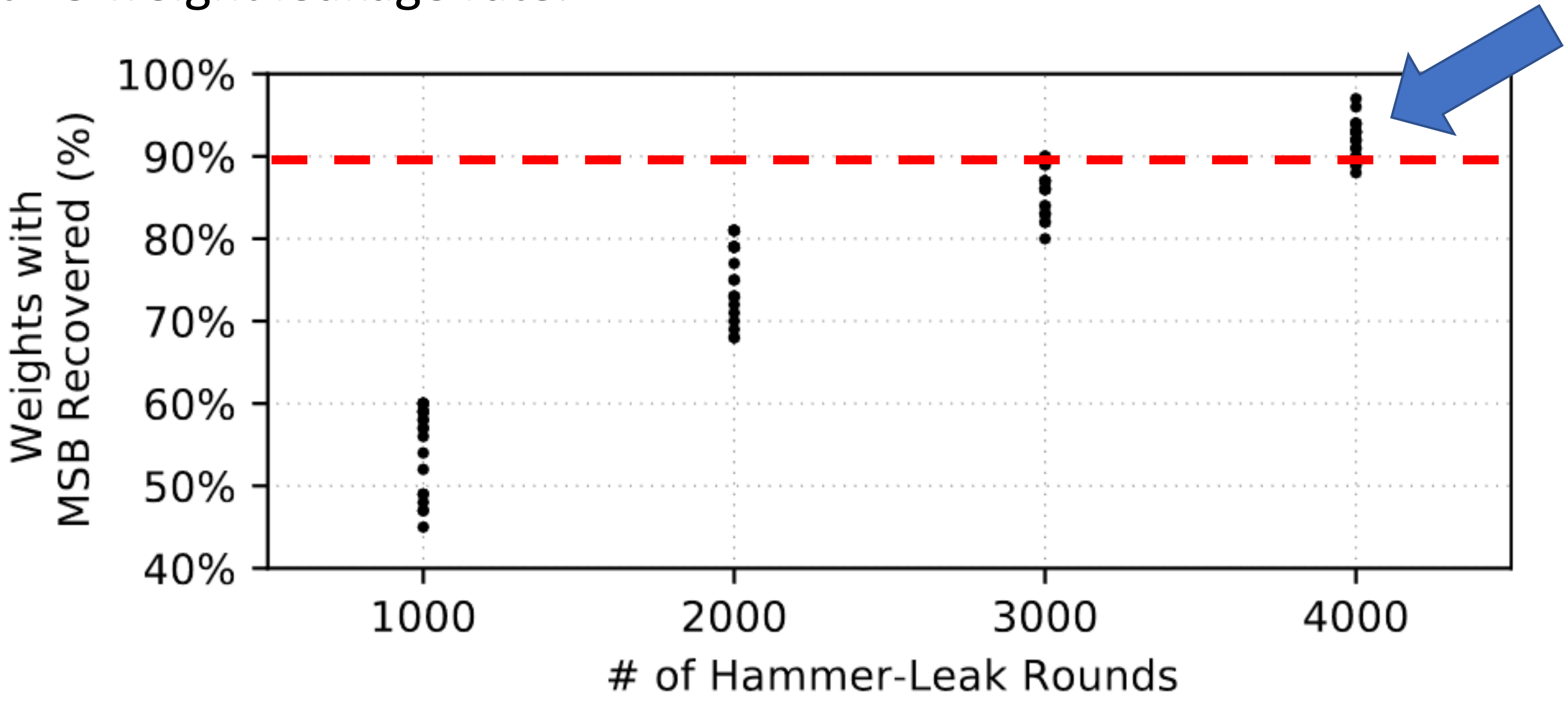


Figure: Distribution of weights with MSB recovered across 21-layers

Results: Mean Clustering Training

- Increasing attack round generates effective substitute model with ***higher accuracy & fidelity***.
- At 4000 rounds, we could achieve similar adversarial example *attack performance as the white-box attack.*

CIFAR-10 (ResNet-18)	**Time (Days)**	**Recovered (MSB) (%)**	**Accuracy (%)**	**Fidelity (%)**	**Adversarial Example Attack (%)**
Architecture Only	-	0	73.18	74.29	61.33
1500 Rounds	3.9	60	76.61	77.56	50.4
3000 Rounds	7.8	80	86.93	88.51	8.13
4000 Rounds	***10.4***	***90***	***89.59***	***91.6***	***1.61***
Best-Case (White Box)	-	100	93.16	100.0	0.0

Comparison with Existing Methods:

Recovery Method	**Accuracy (%)**	**Adversarial Example Attack (%)**
Architecture only e.g., DeepSniffer (ASPLOS 20)	72.68	62.68
DeepSteal (Architecture + Partial Weight-Bit Information)	***90.35***	***1.2***

- DeepSteal shows ***~18 %*** improvement in accuracy compared to the existing remote side-channel attacks which only focus on recovering the architecture only information of DNN.
- Fine-grained bit information significantly improves the adversarial attack performance as well.

Comparison with Existing Methods:

Attack Threat Model	**Adversarial Example Attack (%)**
Black-Box (Transfer Cui et. al.)	20.47
White-Box (PGD Madry et. al.)	0.0
DeepSteal (ours)	***1.2***

- DeepSteal threat model falls in the gray-box zone (architecture known) between white-box and black-box attack.
- Fine-grained bit information achieves almost similar success rate as the white-box attack.

Conclusion:

- DeepSteal with the exploitation of a remote side channel, for the ***first time***, can exfiltrate fine-grained ***weight information*** in bulk from DNN model.

- DeepSteal can recover substitute model with high accuracy and fidelity (~ ***90 %).***

- The adversarial examples generated from the substitute model is as ***effective as a white-box attack.***

- Our proposed attack opens a practical solution to identical model recovery and urges the community to ***invest in future defense solutions.***

Thank You & Questions?

Adnan Siraj Rakin

Email: asrakin@asu.edu

Md Hafizul Islam Chowdhuryy

Email: reyad@knights.ucf.edu

Fan Yao, Ph.D.

Email: Fan.Yao@ucf.edu
Website: https://casrl.ece.ucf.edu

Deliang Fan, Ph.D.

Email: dfan@asu.edu
Website: https://dfan.engineering.asu.edu/

This work is supported in part by the National Science Foundation under Grant No. 2019548, No. 2019536, No. 1931871, and No. 2144751.