Cracking the AI Nut: From Output Extraction to Shuffling Countermeasure

A Side-Channel Evaluation of DNNs Using the CrackNuts Platform

Le Wu[1], Liji Wu[1*], Xiangming Zhang[1], Yuyang Pan[2], and Jian Wu[3]

[1] School of Integrated Circuits, Tsinghua University;

[2] Beijing UnionPay Card Technology Co., Ltd; [3] Beijing Pairui Micro Technology Co., Ltd.

wul22@mails.tsinghua.edu.cn; {lijiwu, zhxm}@tsinghua.edu.cn; panyuyang@qq.com; wujian@ahcx.cloud

Motivation

- Deep Neural Network (DNN) inference on edge devices **exposes AI models to side-channel threats**.
- Beyond the well-studied leakage of inputs or parameters, we **identify output-stage leakage** as a critical yet underexplored vulnerability.
- In this work, we:
  - Use our **self-developed CrackNuts platform**;
  - **Recover NN predictions on *STM32F4*** via Simple Power Analysis (SPA);
  - Introduce a **shuffling-based countermeasure leveraging the Fisher–Yates algorithm.**

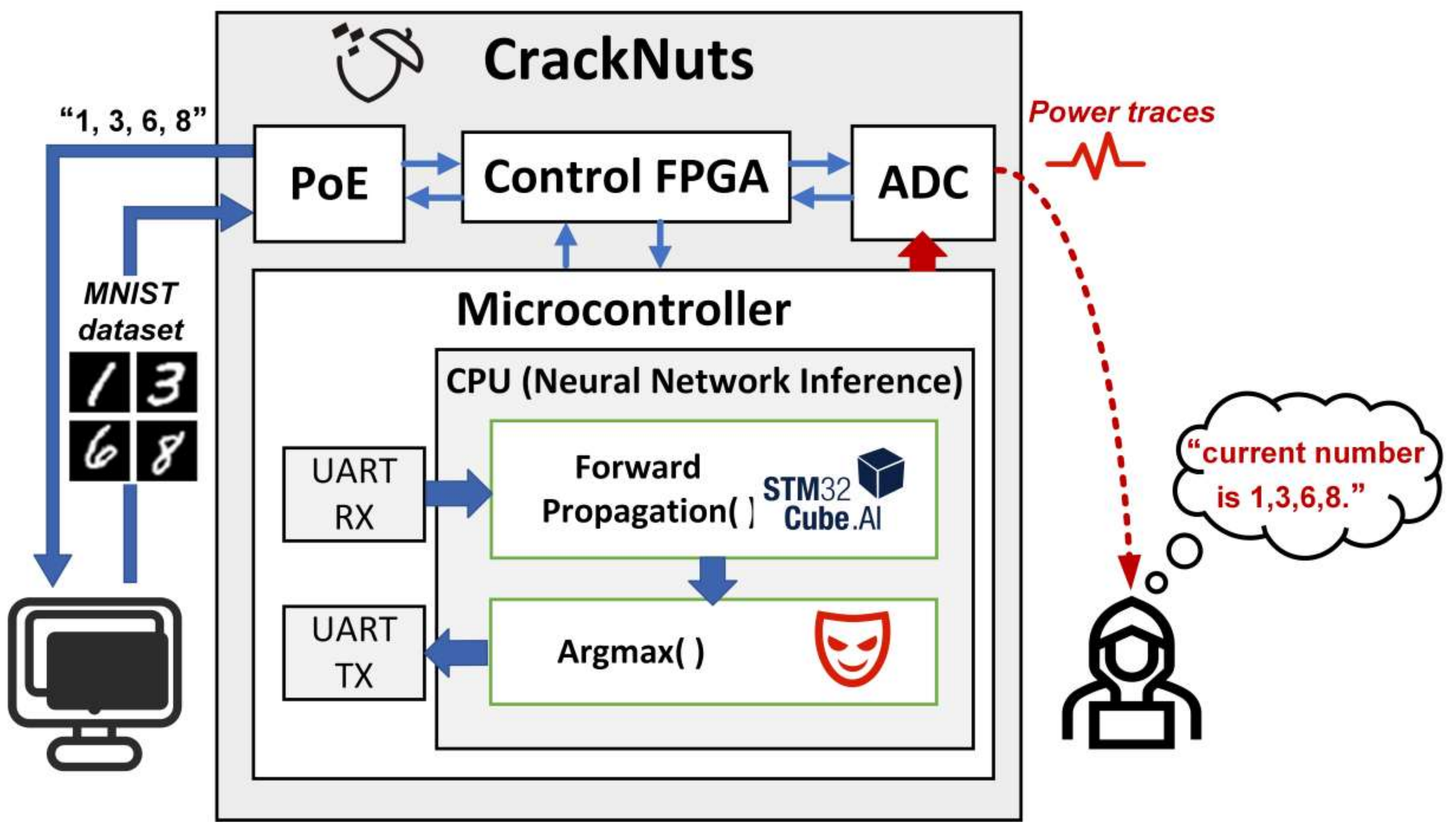

Figure 1. Illustration of the attack setup and model.

Experimental Setup

- We use **CrackNuts, an open-source side-channel security evaluation platform developed by our lab.**

The platform consists of two modular boards:

- **Cracker:** provides power-based side-channel acquisition and control.
- **Nuts:** a swappable MCU board (**STM32F407VG** in this study)

- **Model:**
  - **a DNN trained on the MNIST dataset,** implementing a 10-class handwritten-digit recognition task generated by the X-CUBE-AI tool.

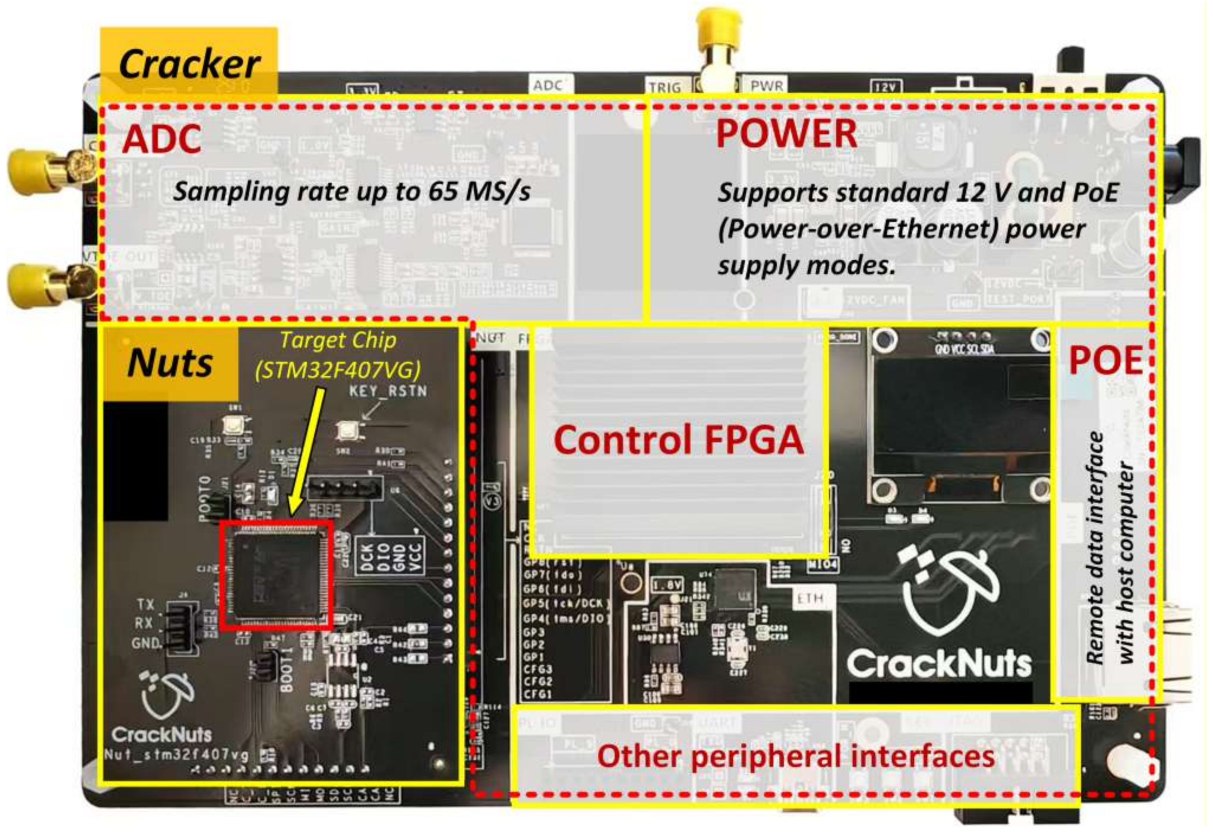

Figure 4. The CrackNuts side-channel evaluation platform.

Future Directions

- Perform reverse engineering of DNN structures via side-channel techniques using CrackNuts.
- Evaluate hardware vulnerabilities of DNNs via fault-injection experiments using CrackNuts.

Methodology

Extract Output via SPA

- **Attack Target:** The *argmax* function identifies the index of the highest-probability output.

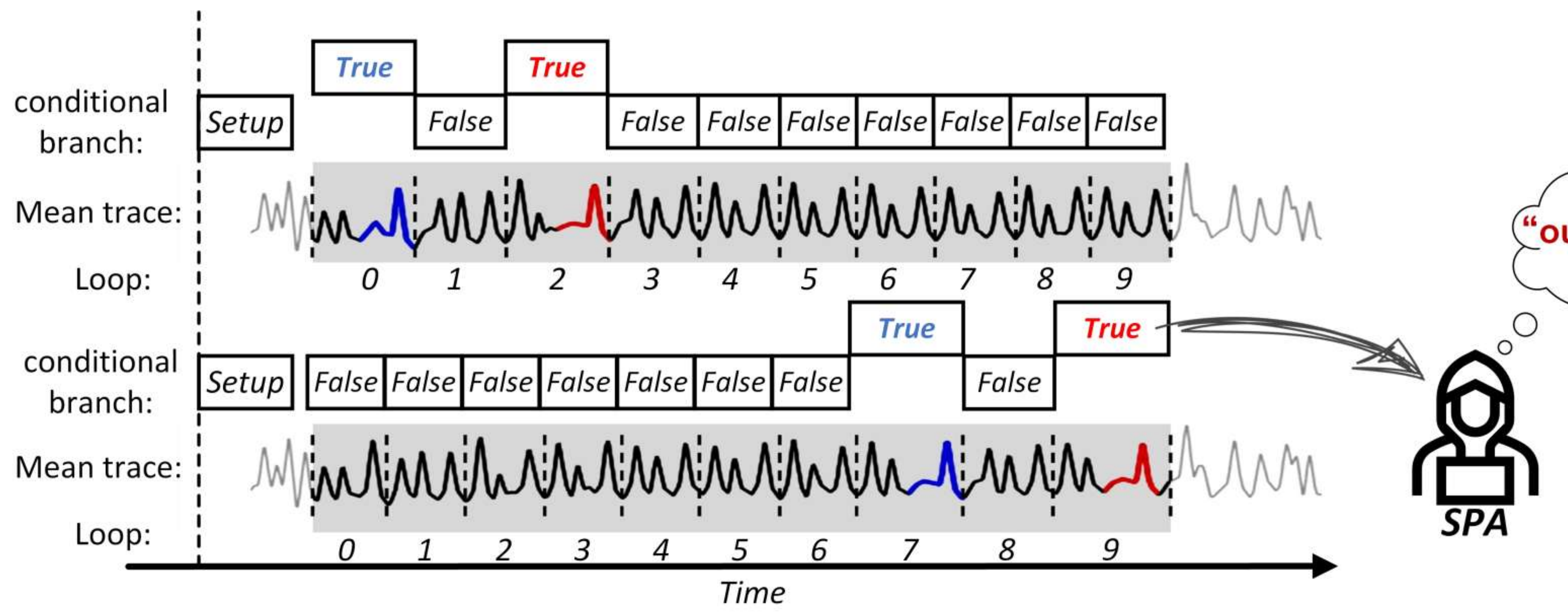

Figure 2. Output Extraction via SPA.

Fisher–Yates–Based Shuffling Countermeasure

- The output indices are randomly permuted before **the *argmax* operation, without affecting the correctness of the final result.**
- Randomization disrupts the correlation between execution order and output values, thereby **obfuscating data-dependent power variations.**

Baseline Neural Network (without Side-Channel Countermeasure)

```
Input: RX_Buffer (data from the buffer of UART_RX)
Output: count (predicted label)
N = 10

UINT_TO_FLOAT(RX_Buffer, Din, 28×28)
NEURAL NETWORK(Din, Dout)

max ← 0
count ← 0
for i = 0 to N−1 do
    if Dout[i] > max then
        max ← Dout[i]
        count ← i
    end if                    Generate power leakage
end for
```

Neural Network with Shuffling Countermeasure

```
Input: RX_Buffer (data from the buffer of UART_RX)
Output: count (predicted label)
N = 10

UINT_TO_FLOAT(RX_Buffer, Din, 28×28)
NEURAL NETWORK(Din, Dout)
SHUFFLE_INDICES(shuffled_index, N)

max ← 0
count ← 0
for i = 0 to N−1 do
    idx ← shuffled_index[i]
    if Dout[idx] > max then
        max ← Dout[idx]
        count ← idx
    end if
end for
```

→ *Fisher–Yates Shuffle*

```
function SHUFFLE_INDICES(idx_array, size)
    for i = 0 to size−1 do
        idx_array[i] ← i
    end for

    for i = size−1 to 1 do
        j ← random integer in [0,i]
        Swap idx_array[i] ↔ idx_array[j]
    end for
end function
```

Figure 3. Pseudocode of neural-network inference on the MCU.

Results

The neural-network output classification process induces distinct power patterns due to conditional branching behavior:

- **The red-highlighted region of the waveform** indicates the final update of the maximum value during the comparison operation (the attack point);
- **The blue-highlighted waveform segments** correspond to earlier updates that are not relevant to the attack.

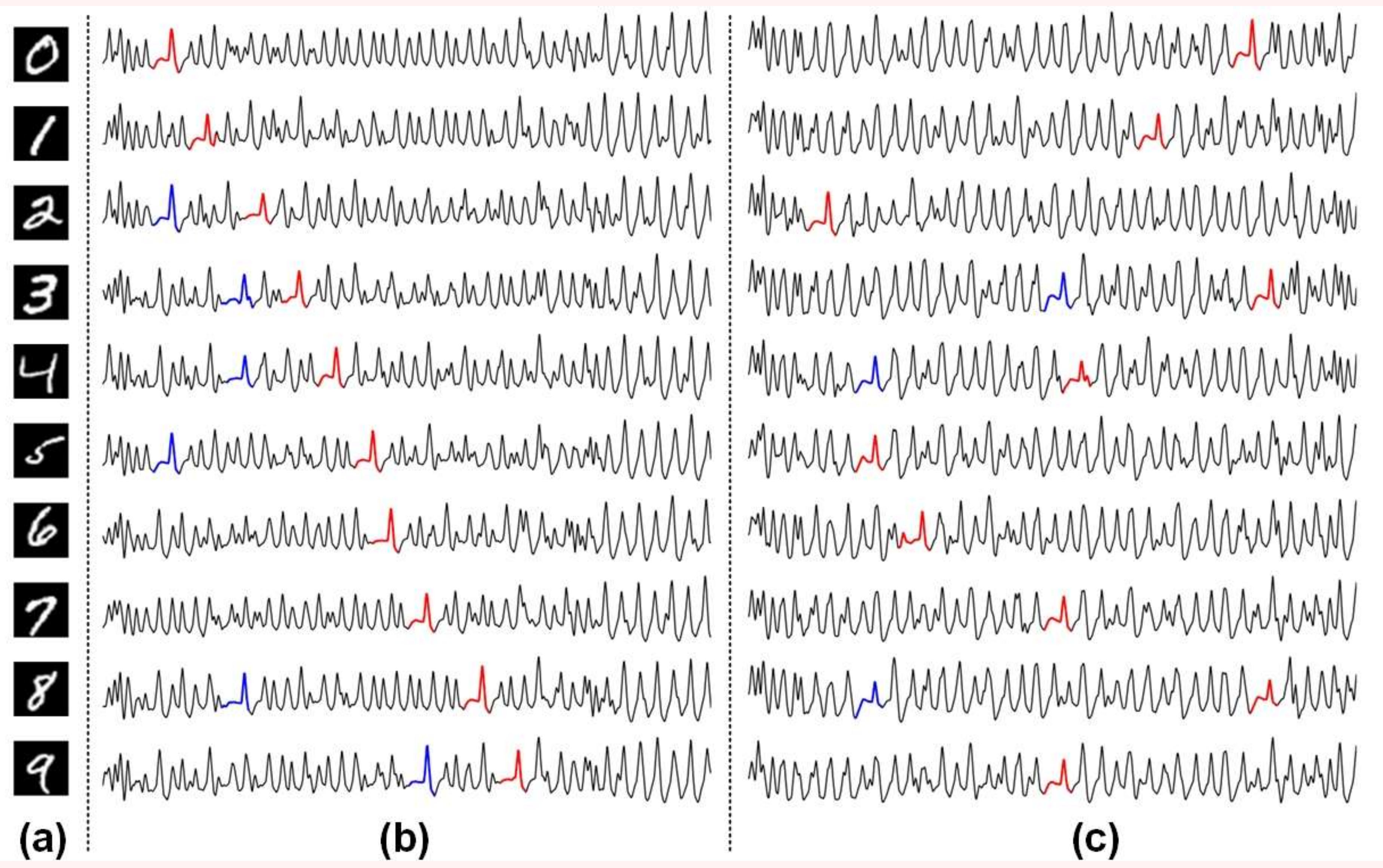

Figure 5. Power traces of neural-network output classification.
(a) Input image; (b) Without countermeasure; (c) With shuffling-based countermeasure.

Acknowledgements and References

[1] Méndez Real M, et al., "Physical side-channel attacks on embedded neural networks: A survey". *Applied Sciences*, 2021, 11(15): 6790.

[2] Maji S, et al., "Leaky nets: Recovering embedded neural network models ⋯". *IEEE Internet of Things Journal*, 2021, 8(15): 12079-12092.

[3] Puškáč L, et al., "Make Shuffling Great Again ⋯". *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2025, 7(33): 1959-1971.

[4] Brosch M, et al., "Counteract side-channel analysis of neural networks by shuffling". *DATE 2022*. IEEE, 2022: 1305-1310.