Securing ML models by encrypting their weights and biases using keys derived from PUFs

Literature Review
Date: 19th March 2025

Nupur Dhananjay Patil
IMT2022520

Paper:
Securing Binarized Neural Networks via PUF-Based Key Management in Memristive Crossbar Arrays

Paper summary:

- Binarized Neural Networks (BNNs) implemented on memristive crossbars are vulnerable to theft attacks since their parameters can be physically extracted(non-volatile)
- The paper proposes using PUFs to secure BNN parameters by transforming weights before storing them in the crossbar.
- Two key protection methods are introduced – inversion and swapping
- These transformations make stolen models unusable without the correct PUF key while maintaining low hardware overhead.
- This can be extended to other deep learning models.

- BNNs - weights and activations take values as +1 and −1 only. Why BNN - for edge applications because its binary computation during inference reduces computational complexities and saves energy.
- It is stored on the RRAM crossbar array by mapping its binary weights (+1, -1) to different resistance states of memristors.
- **Why RRAM crossbars? -** enable in-memory computing. The risk arises because an attacker with physical access can extract the stored resistance values.
- A single binary state is represented by 2 RRAM cells in columns, for "+1" the top cell is programmed to LRS and bottom to HRS and vice-versa.
- In the paper they have given a methodology to implement BNN. The Batch Normalization (BN) layer is combined with the Matrix-Vector Multiplication (MVM) layer, and a comparator decides the output.

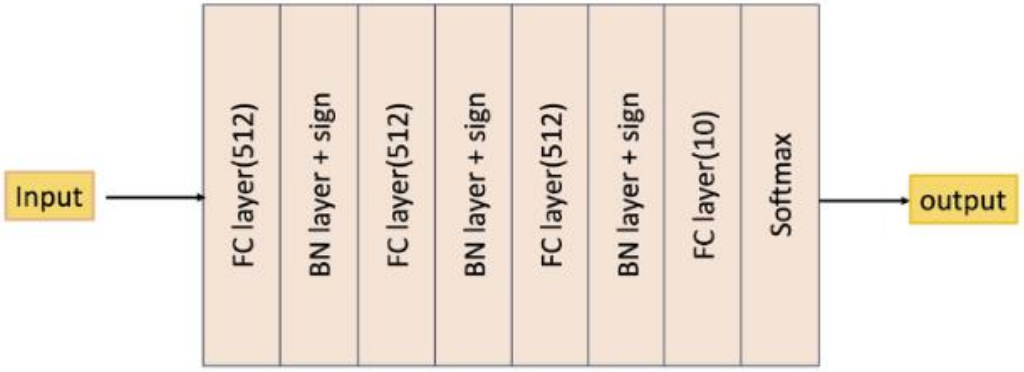
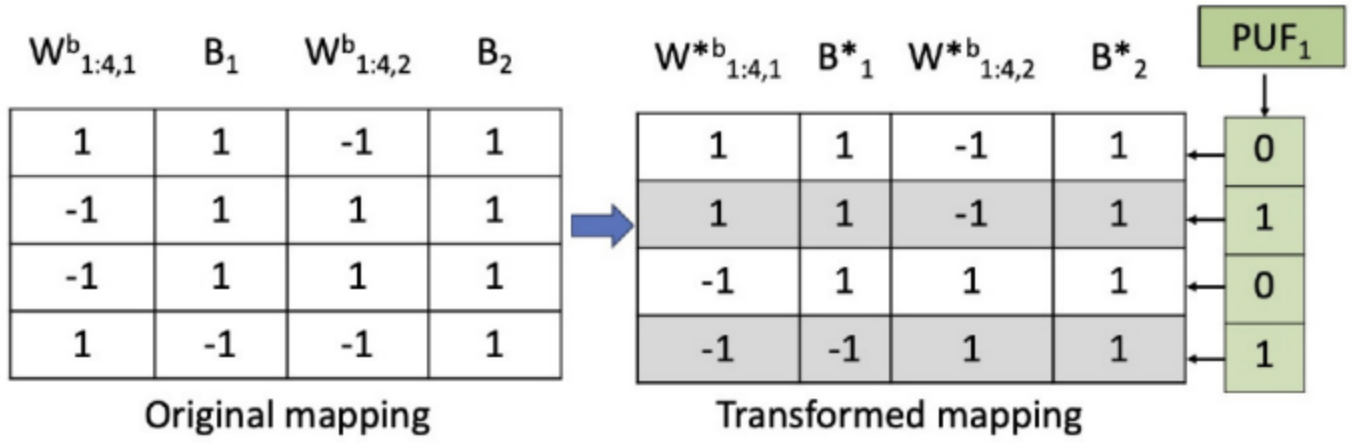


Fig. 1. Architecture of neural network trained used for MNIST dataset.

- The attacker can access the edge hardware with an RRAM crossbar, it knows the topology and can read the conductance states of RRAM devices using micro-probing (methods for the same are mentioned in the reference).
- transforming the "W b j,k" into a new matrix "W∗ b j,k" with a PUF response and then storing it in the crossbar. The length of the PUF response required for row and column inversion equals the row size and column size resp.
- The paper uses a Parity RRAM PUF based on a 1T1R (one transistor, one resistor) crossbar. This PUF was previously proposed by the authors and can also be extended to work as a random number generator.

1. Row inversion:

Main logic - Each row of the BNN weight matrix is flipped (multiplied by -1) based on a PUF-generated secret key. If the key for a row is 1, all weights in that row are inverted; if the key is 0, the weights stay the same.



XOR gate is used to ensure that the inverted weights and inputs are correctly processed during inference. Since the row inversion method flips the weights based on the PUF key, we must also adjust the input signals so the model behaves as expected.

Inference Logic

The same PUF key used during training is applied again.

The XOR gate is used to check the key:

- If PUF key = 1, the input is also inverted (multiplied by -1) before being sent to the crossbar.
- If PUF key = 0, the input remains the same.

Since both the weights and the inputs are inverted together, the final multiplication results cancel out, ensuring the model produces the correct output.

INFERENCE ACCURACY WITH ROW INVERSION

Transformed layers	PUF key length	Inference accuracy without right PUF key
None (without PUF protection)	0	96.74%
FC 1	784	9.83%
FC 2	512	8.28%
FC 3	512	13.49%
FC 1,2,3	$784 + 2 \times 512$	11.13%

- In column inversion – only difference is both fully connected layer weights *and* the Batch Normalization (BN) parameters are transformed are transformed in a similar fashion.

INFERENCE ACCURACY WITH COLUMN INVERSION

Transformed layers	PUF key length	Inference accuracy without right PUF key
None (without PUF protection)	0	96.74%
FC +BN 1	512	10.71%
FC +BN 2	512	7.35%
FC +BN 3	512	13.8%
FC + BN 1,2,3	3×512	10.2%

- Inversion techniques are applied to the weight matrices of these layers.
- They applied 10 different PUF keys per layer to test how secure and reliable the transformation method is. The idea is that by using different keys across layers, attackers won't be able to reconstruct the model unless they have all the correct PUF key.
- They also performed a power analysis and concluded that the overhead incurred by the inversion logic (XOR gate) is negligible (less than 1% increase in power).

Method 2 : Swapping

- A PUF-generated key is assigned to each column (or row) of the weight matrix.
- Swapping Process : transforming the weight matrix together with the BN layer values by locally swapping the values along the rows or columns based on the PUF response. They have used column swapping.
- If the PUF key = 1, two adjacent weights are swapped; if PUF key = 0, weights remain unchanged.
- The scrambled weight matrix (after swapping) is stored in the RRAM crossbar. During inference, the same PUF key is used to swap back the weights, restoring the correct model.

$$R_k = \begin{cases} 1, \text{ then } B_k^* = B_{k+1} \text{ and } B_{k+1}^* = B_k \\ 0, \text{ then } B_k^* = B_k \text{ and } B_{k+1}^* = B_{k+1}. \end{cases} \qquad R_k = \begin{cases} 1, \text{ then } W_{j,k}^{*b} = W_{j,k+1}^b \text{ and } W_{j,k+1}^{*b} = W_{j,k}^b \\ 0, \text{ then } W_{j,k}^{*b} = W_{j,k}^b \text{ and } W_{j,k+1}^{*b} = W_{j,k+1}^b \end{cases}$$

- Swapping weights in one layer does not impact accuracy much, but swapping all layers reduces accuracy to 52.96%.
- The method uses three PUF keys, each 256 bits long, which is half the column size of the weight matrix.
- The drop in accuracy without the PUF key does not mean the model is weak; it just shows that the transformed weights alone don't work correctly.

INFERENCE ACCURACY WITH SWAPPED COLUMNS

Transformed layers	PUF key length	Inference accuracy without right PUF key
None (without PUF protection)	0	96.74%
FC +BN 1	256	79.79%
FC +BN 2	256	88.4%
FC +BN 3	256	94.57%
FC + BN 1,2,3	3×256	52.96%

Ending remarks:

The inversion scheme is more secure because it requires a longer key, but the swapping scheme can still provide strong protection, especially for larger weight matrices.

Swapping is better for larger weight matrices because it requires shorter PUF keys compared to inversion. Since key length affects security, using shorter keys for large matrices still provides strong protection while reducing hardware complexity.

Thus, if the PUF key length is limited, swapping is a better choice.

A hybrid approach can be used: inversion for smaller matrices and swapping for larger ones.

The proposed row inversion and swapping techniques can also be extended to RRAM-based multibit DNN implementations, which can be future work.