Recovering AES keys with a deep Cold boot attack

I. Zimerman* [1], *E.Nachmani*[*1 2], *L.Wolf* [1]
* Equal Contribution

1 Tel Aviv University

2 Facebook AI Research

Introduction

- In this work, we present a deep neural network based solution for implementing the **Cold Boot Attack** on AES keys

- Outline:
    - Cold Boot Attack
    - Our Method
    - Empirical Evaluation & Ablation Study
    - Conclusions

Cold Boot Attack

- Cold boot attack is a side channel attack for stealing encryption keys

- The attack is based on two assumptions:
    1. The key has some fixed known redundancy
    2. The attacker has access to a corrupted key

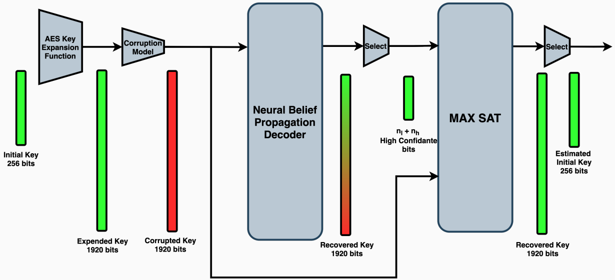
- The practicality of these assumptions

The Computational Problem

- We focus on the problem of recovering an encryption key from its corrupted key by using the redundancy
- Previous techniques were based on:
  - Integer programming
  - Techniques from the field of error correcting codes
  - SAT and MAX-SAT solvers
- We focus on the AES cipher
- Our method: Use deep neural network to approximate the key, and use it to target the SAT solver

Our Method

- Our architecture contains two components:
  - A Neural belief propagation decoder with neural S-box layers
  - A Partial MAX-SAT solver



Approximate the Key with Neural Network

- Empirically, vanilla neural networks fails in this area

- Inspired by deep methods for error correcting codes we decided to use Message Passing Neural Network

- By using a new formulation of the key expansion function as liner error correcting code we success to define an appropriate deep architecture

- With the new formulation, we can use known methods, such as Belief Propagation Neural Network (Nachmani et al.)

Formalize the AES Key Expansion as a computational graph

- The AES key expansion defined by:

$$w_i = w_{i-k} \oplus S(R(W_{i-1})) \oplus c_i \tag{1}$$

$$w_i = w_{i-k} \oplus S(W_{i-1}) \tag{2}$$

$$w_i = w_{i-k} \oplus w_{i-1} \tag{3}$$

  Where W_i is the i dword in the key , S is a S-box transformation, R is a rotation function, and c_i, k are some constants

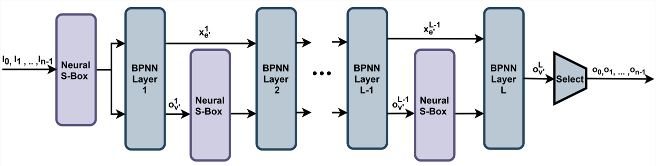
- Although S is nonlinear, by adding to W the variables $S(R(W_{i-1}))$ and $S(W_{i-1})$, one can convert this equations to a linear form such that $HW = 0$

Taylor the Belief Propagation Neural Network

- The Belief Propagation Neural Network defined by a parity check matrix H

- H non contains the nonlinear constrains (S-box constrains)

- Exploiting the nonlinear constraints with the neural S-box layers after each original layer



The S-box layer and the Neural S-box

- Design to exploit the nonlinear constrains
- Each s-box layer consist of neural S-box instances
- Extend the S-box transformation behaviour for fraction values by a fully connected neural network
- Despite S is highly non-linear, not differentiable and designed to be resistant to such attacks our results show the effectiveness of this tool

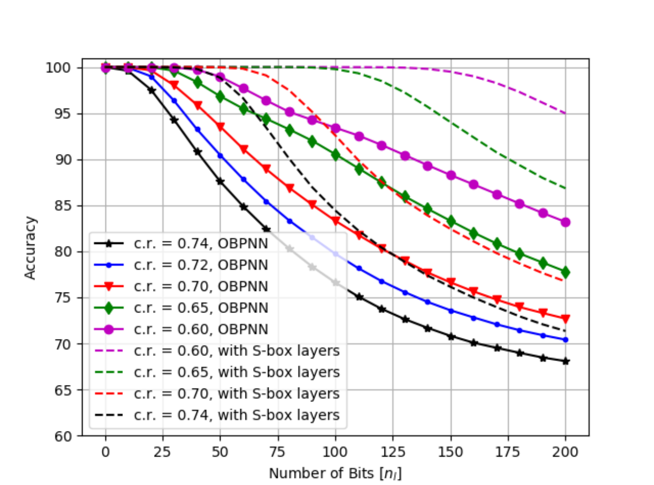
Results

Table 1: Performance evaluation for theoretical model ($\delta_1 = 0$). The success rate of cold boot attack for AES-256 with different corruption rates. Higher is better.

Model/Corruption rate	60%	65%	68%	70%	72%	74%
Tsow et al.	**100.0**	**100.0**	N/A	0.0	0.0	N/A
MAX-SAT	97.92	93.95	84.12	73.56	49.53	15.95
Ours	99.51	97.05	**91.20**	**84.10**	**54.52**	**22.35**

Ablations

- To isolate the contribution of the neural-sbox, we use two ablations:
    1. LC: an architecture that only defined by the linear constrains
    2. OBPNN: an architecture that not include neural s-boxes



Conclusions

- ML is often considered unsuitable for problems in cryptography, we present convincing evidence in support of employing deep learning in this domain
- We successfully approximate the S-box transformation by a neural network, and the ablations study emphasizes the power of this tool
- A new error correcting code representation of the AES family of codes
- Combine the approach of the error correcting codes with the SAT solver approach to achieve SOTA results