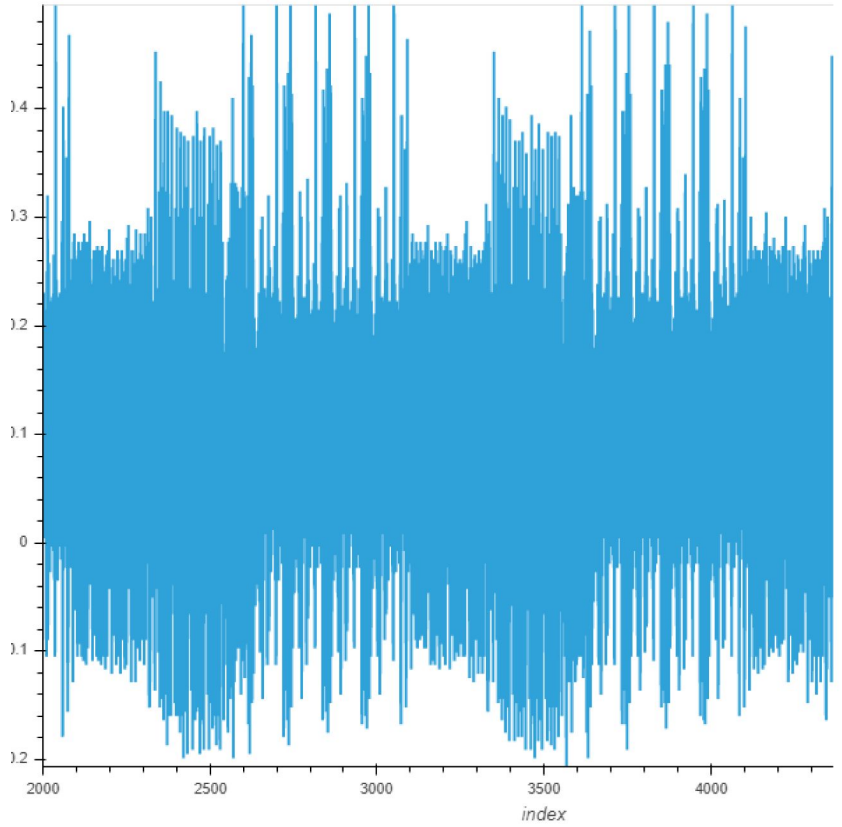
Using Machine Learning to Assist Hardware Attacks

Soner Seckiner, Amelia Kwak, Jacob Sepinuck

Problem

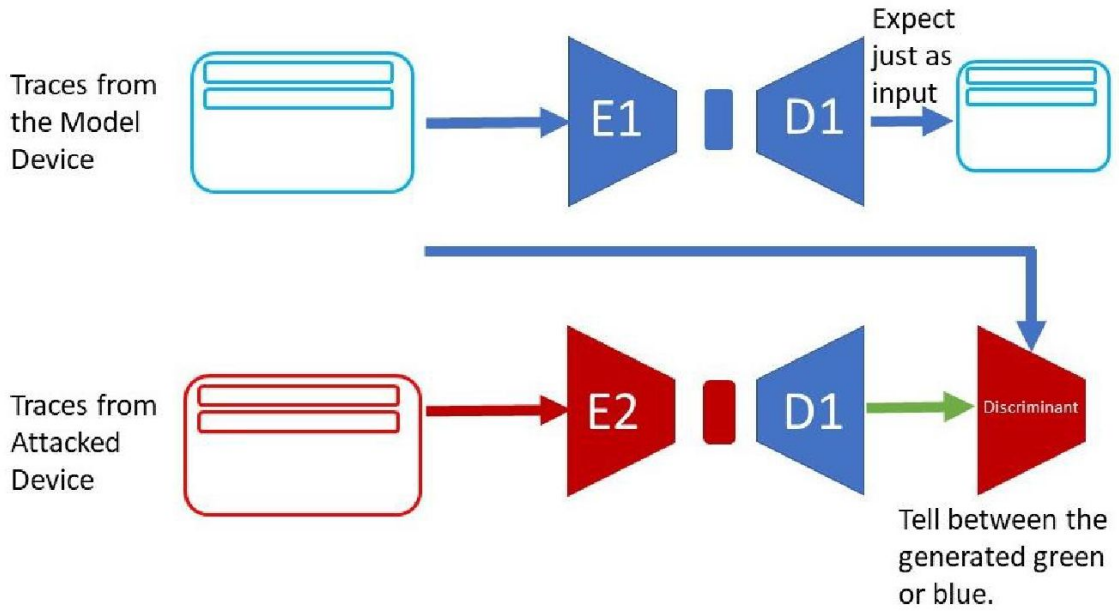
- Cryptographic devices perform encryption that is dependent on the installed hardware
- Such devices emit ‘side channels’ (i.e., power draw, heat, etc.) which can give an attacker information about the chip
- Attackers can collect side-channel information (usually power traces since they are easy to measure and less noisy) and use mathematical models to correlate the power draw with encryption keys
- Issue: These power traces can be heavily dependent on the method of measurement and can be disguised (masked) by the device being attacked. This makes it harder for the attacker to find the encryption key.



Sample power traces taken from cryptochip

Approach

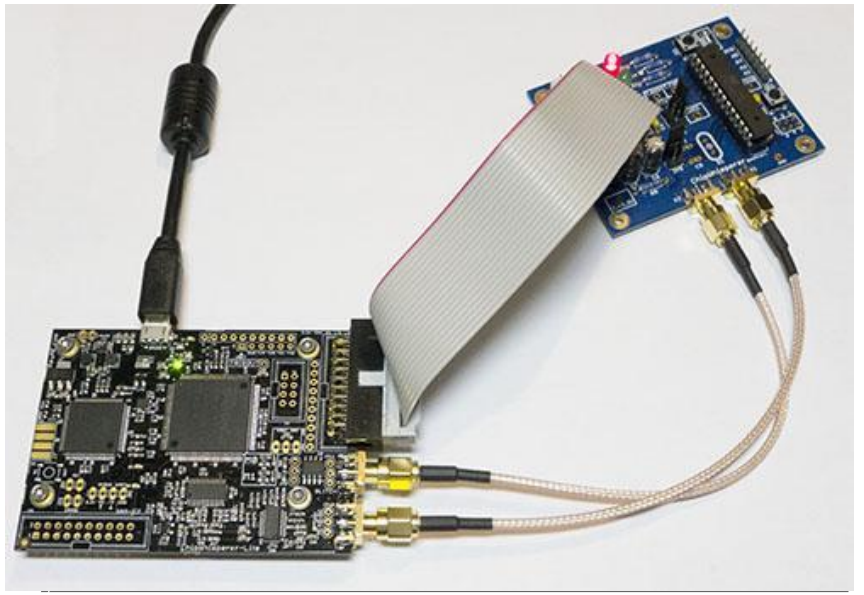
- We utilized machine learning to simplify masked side-channel measurements in order to make it easier for attackers to determine encryption keys of masked signals.
- A GAN (Generative Adversarial Network) autoencoder (AE) is trained on a model device. A discriminator evaluates traces from both the model and attacked device and uses back-propagation to update the attacked device's encoder to generate signals comparable to the model signals. This process is continued until the autoencoder can produce traces from the attack signals that are indistinguishable from the model signals.

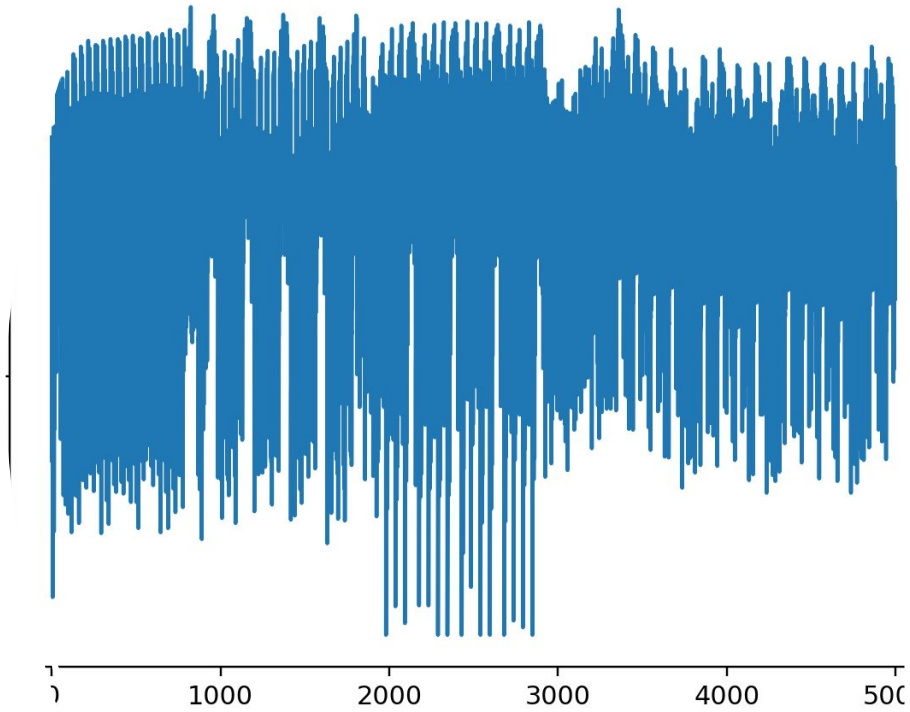


Preparation for Data Collection

- We used the same device that the ASCAD database uses
- The device runs 128-bit AES encryption
- We learned the secret key off the ASCAD database and applied it to our Chipwhisperer device
- **ASCAD database:** Masked (i.e. protected) 128 bit AES.
- **Our device:** Unmasked 128 bit AES.

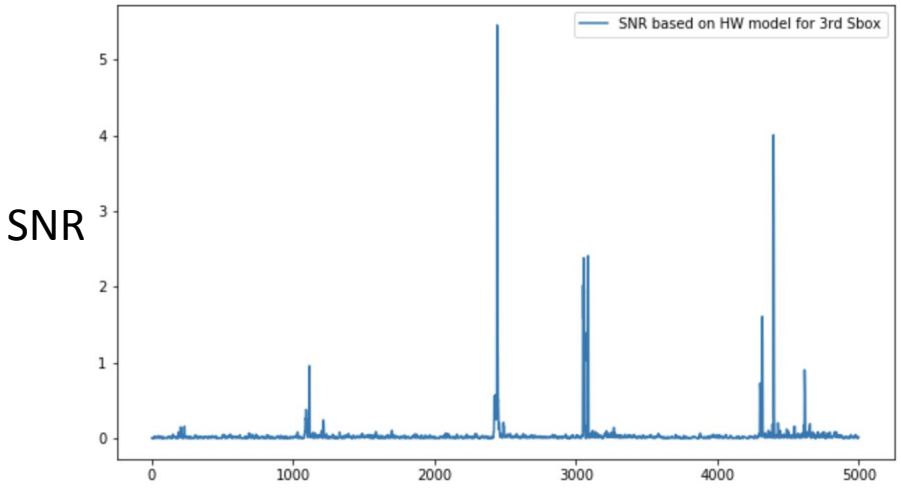
Online: https://github.com/ANSSI-FR/ASCAD



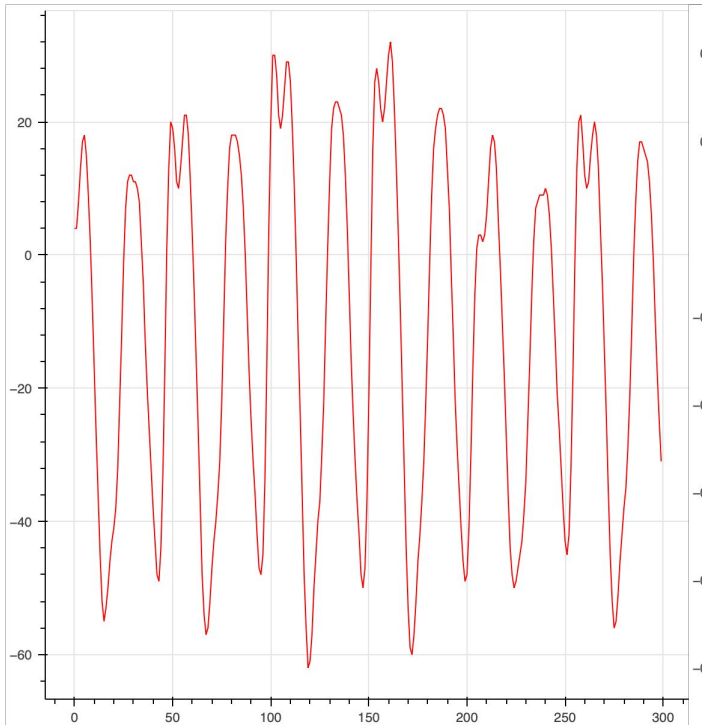


Data Collection & Preprocessing

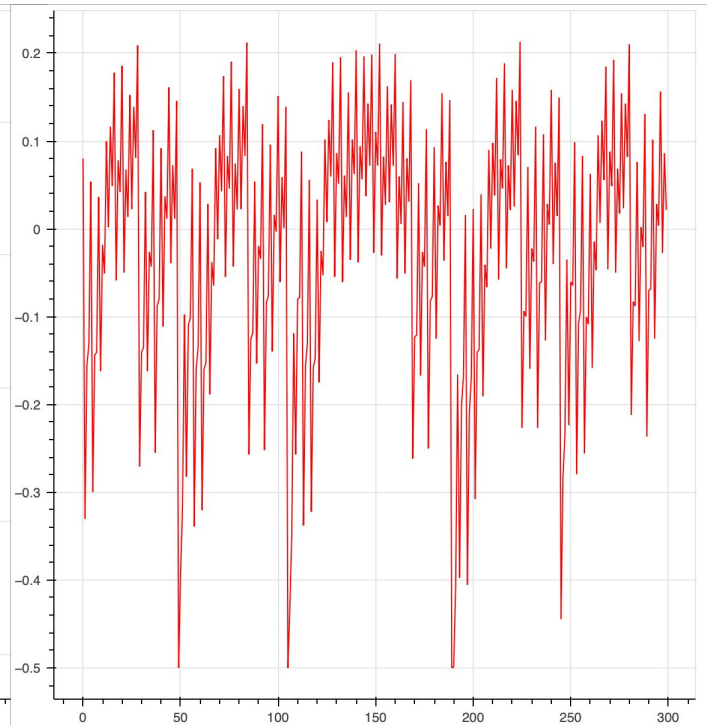
- Collected 100,000 unmasked AES traces from the same device with the same cryptographic keys as those used in the masked AES database (ASCAD) where each trace consists of 5000 samples
- Identified Signal-to-Noise Ratio of 3rd S-Box
- S-Box: basic component of encryption key algorithms
- Applied Hamming Weight of the 3rd S-box because the signal is highly correlated with Hamming Weight
- Identified the range of the 3rd S-Box and trimmed traces to this range because our reference database ASCAD published the same 3rd S-Box signals
    - This will be the data that the GAN is trained/tested on



3rd S-Box Identification



Masked AES ASCAD 3rd S-Box



Unmasked AES Collected 3rd S-Box

Validation of the Collected Data with Side-Channel Attack

- Key stored in ASCAD database:

0x4d,0xfb,0xe0,0xf2,0x72,0x21,0xfe,0x10,

0xa7,0x8d,0x4a,0xdc,0x8e,0x49,0x04,0x69

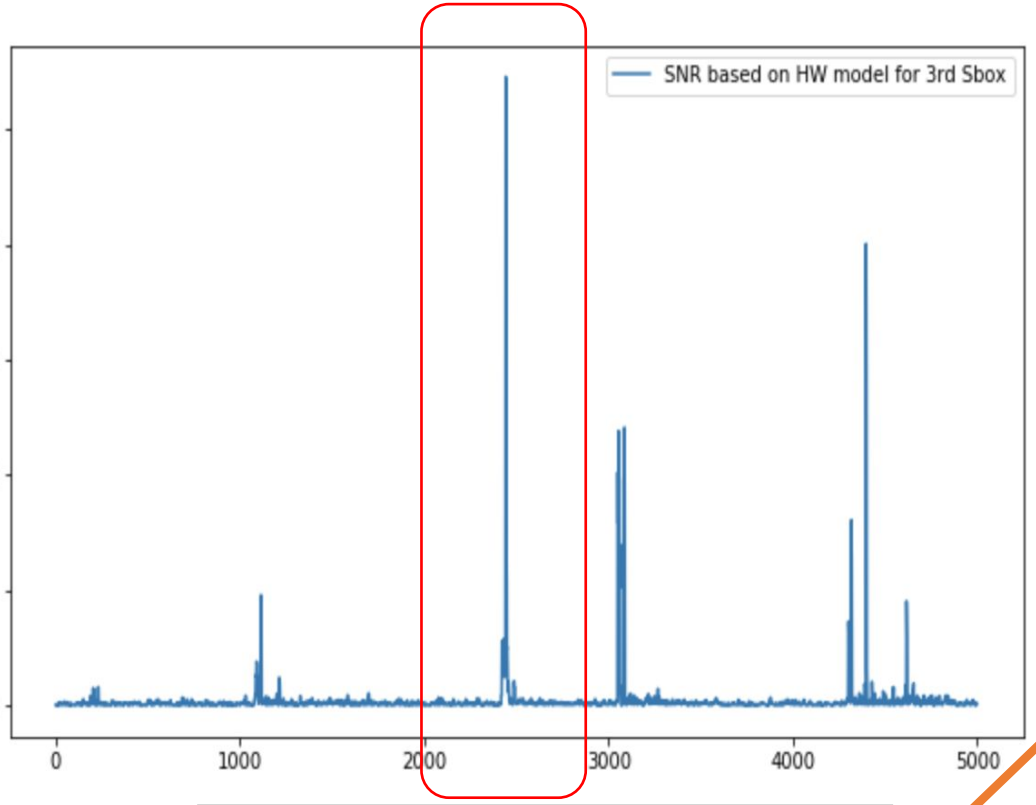
- We used Correlation Power Analysis to validate the key of the collected data
- Hamming Weight is used as a power model
- Highest correlation is presented, and its calculated key is presented

```
leak_model = cwa.leakage_models.sbox_output

attack = cwa.cpa(proj, leak_model)

results = attack.run()
```

```
print(results)
```

```
Subkey KGuess Correlation
  00     0x4D    0.26777
  01     0xFB    0.29003
  02     0xE0    0.25682
  03     0xF2    0.25862
  04     0x72    0.25381
  05     0x21    0.30222
  06     0xFE    0.24523
  07     0x10    0.25612
  08     0xA7    0.25570
  09     0x8D    0.29639
  10     0x4A    0.26180
  11     0xDC    0.29614
  12     0x8E    0.25300
  13     0x49    0.25942
  14     0x04    0.26382
  15     0x69    0.29467
```

Example from Side-Channel Attack Framework



Target 3rd S-box location

```
    bestguess[bnum] = np.argmax(maxcpa)
    cparefs[bnum] = np.argsort(maxcpa)[::-1]

print("Best Key Guess: ", end="")
for b in bestguess: print("%02x " % b, end="")
```

```
Attacking subkeys: 100%|██████████| 1/1 [00:10<00:00, 10.56s/it]

Best Key Guess: 00 00 e0 00 00 00 00 00 00 00 00 00 00 00 00 00
```

```
print("Known Key: ", end="")
for b in knownkey: print("%02x " % b, end="")
```

```
Known Key: 4d fb e0 f2 72 21 fe 10 a7 8d 4a dc 8e 49 04 69
```

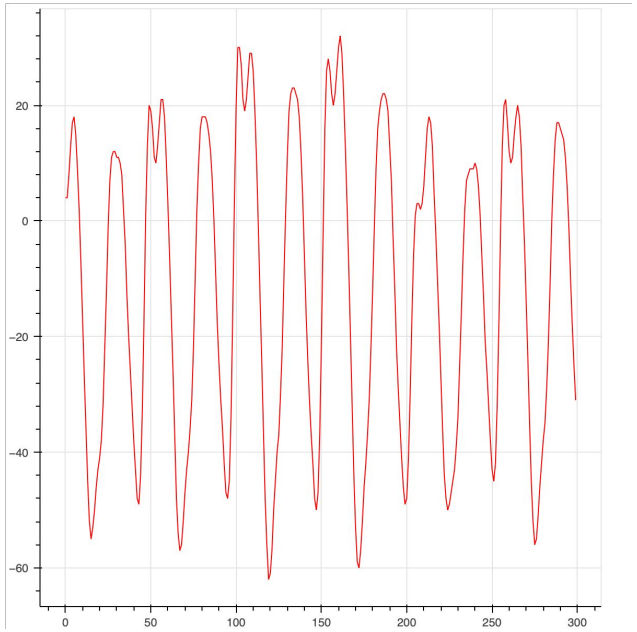
Minimum Ranking Results with Correlation Power Analysis

- We implemented the four different models noted in the table
- The rank corresponds to how close the predicted key is to an 8-bit encryption key
- The lower the rank is to 1, the closer we are to predicting the correct keys
- The Autoencoder network achieved the most accurate prediction of the four networks we tested
    - GAN - AE is capable of predicting the key of some traces

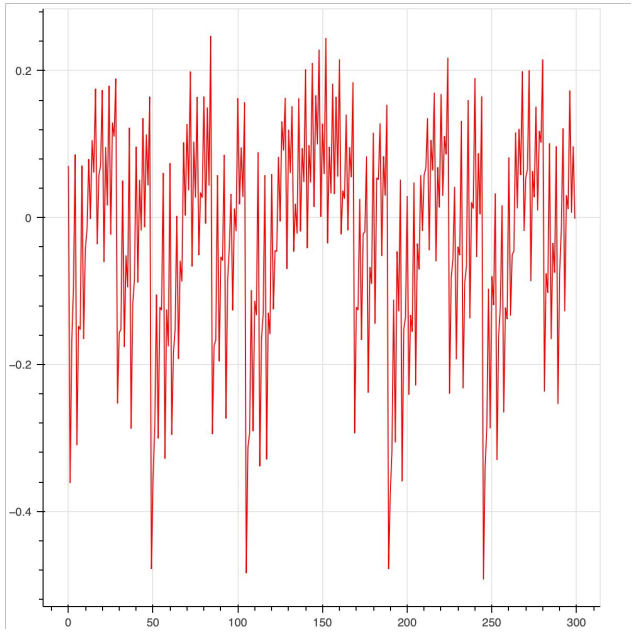
Network	Best Ranking – Lower is better
GAN – MLP	10
GAN – AE	1
GAN – CNN	6
CNN	10

Samples from Transformed Traces

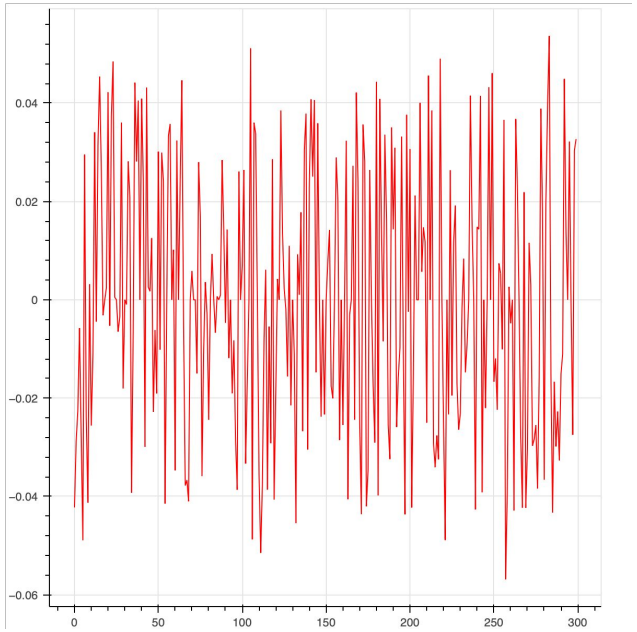
BEFORE



AFTER



GAN – MLP



GAN – AE

Future Plans

- Decrease GAN-AE variability in key prediction
- Expand the framework for other devices
- Expand the framework for other type of countermeasures
- Propose a robust denoising framework

References

- [1] Online: https://github.com/ANSSI-FR/ASCAD
- [2] Online: https://github.com/newaetech/chipwhisperer
- [3] Wu, Lichao, and Stjepan Picek. "Remove some noise: On pre-processing of side-channel measurements with autoencoders." IACR Transactions on Cryptographic Hardware and Embedded Systems (2020): 389-415.
- [4] Wang, Ping, et al. "Enhancing the performance of practical profiling side-channel attacks using conditional generative adversarial networks." arXiv preprint arXiv:2007.05285 (2020).
- [5] Zhang, Ziyue, A. Adam Ding, and Yunsi Fei. "A fast and accurate guessing entropy estimation algorithm for full-key recovery." IACR Transactions on Cryptographic Hardware and Embedded Systems (2020): 26-48.

Thank You

- Questions ?