Google

Deep-Cryptanalysis

Fashion or Revolution?

Elie Bursztein
Google, @elie

with the help of **many** Googlers

August 2021

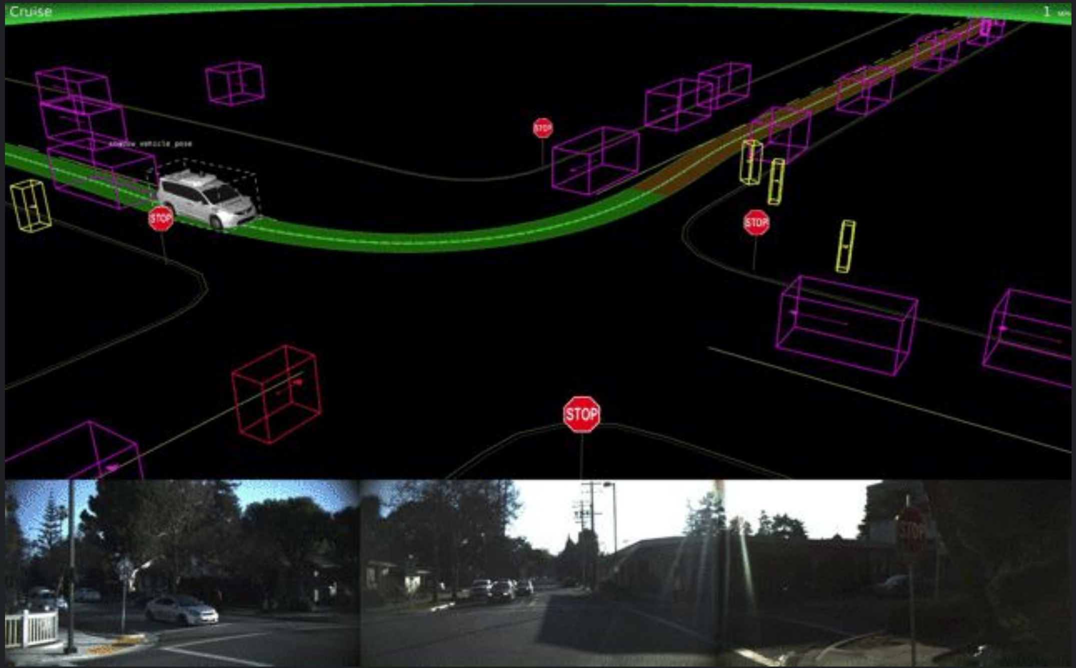
Security and Privacy Group

Slides

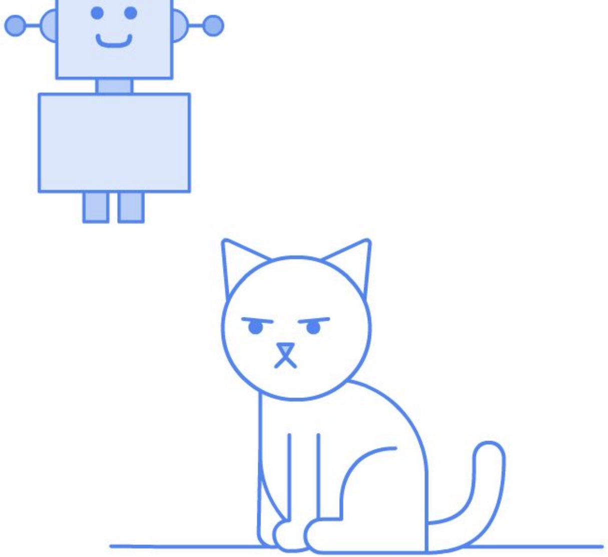
https://elie.net/deep-crypto

Google

Security and Privacy Group



AI is revolutionizing the world



I do crypto research - not AI

Many breakthroughs were made by leveraging AI advances in adjacent fields

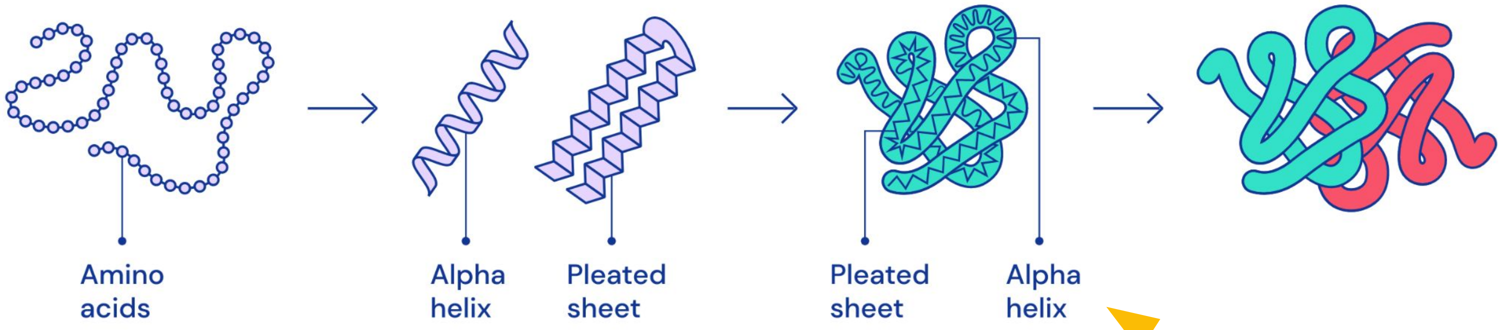
Protein folding problem

Every protein is made up of a sequence of amino acids bonded together

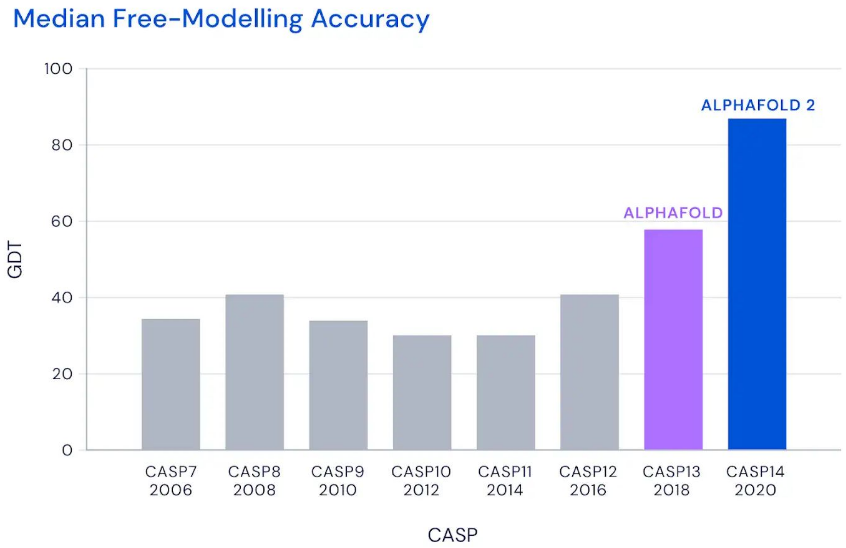
These amino acids interact locally to form shapes like helices and sheets

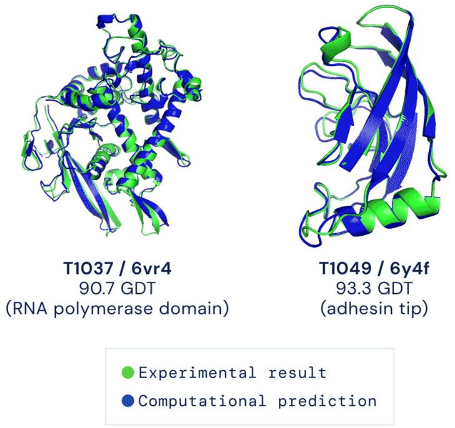
These shapes fold up on larger scales to form the full three-dimensional protein structure

Proteins can interact with other proteins, performing functions such as signalling and transcribing DNA

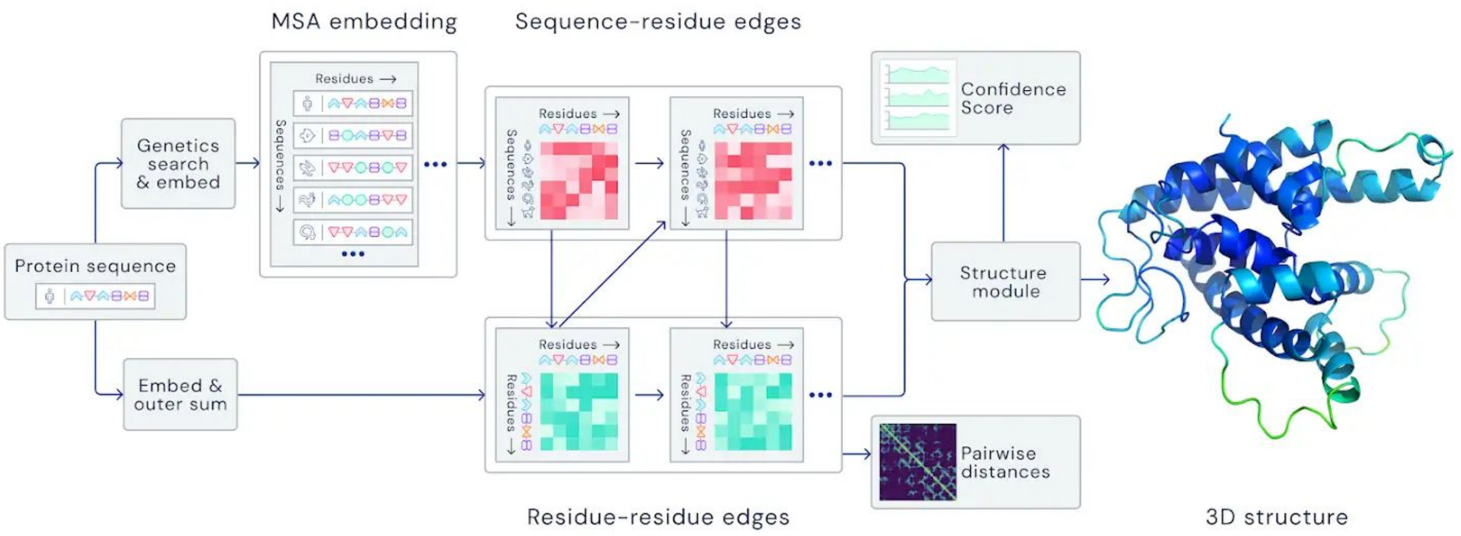


Hard to predict





Alpha Fold2: Resolving structure with atomic precision



Alpha Fold 2 leverages multi-head attention mechanisms originally developed in the NLP field

Where deep-learning advances can benefit cryptanalysis?

Agenda

- Improve existing use-cases
- Not suitable use-cases
- Unlock new use-cases
- Special tactics

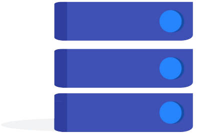
Disclaimer

Applying deep-learning to cryptanalysis is hard as it requires a lot of diverse expertise, fail often, and is technically challenging to implement.

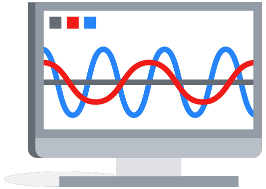
Those difficulties will overtime be greatly reduce as the field will eventually be figured out.

Improving Existing use-cases

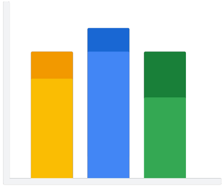
What makes a good target?



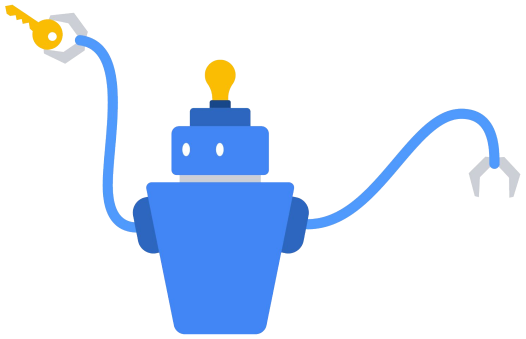
Ability to collect a lot of data



Problem as a lot of structure and is differentiable



It’s about exploiting statistical biases



Side Channel Attacks Automated with Machine Learning

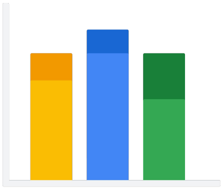
Google

Security and Privacy Group

Template attack on steroids

No trace processing

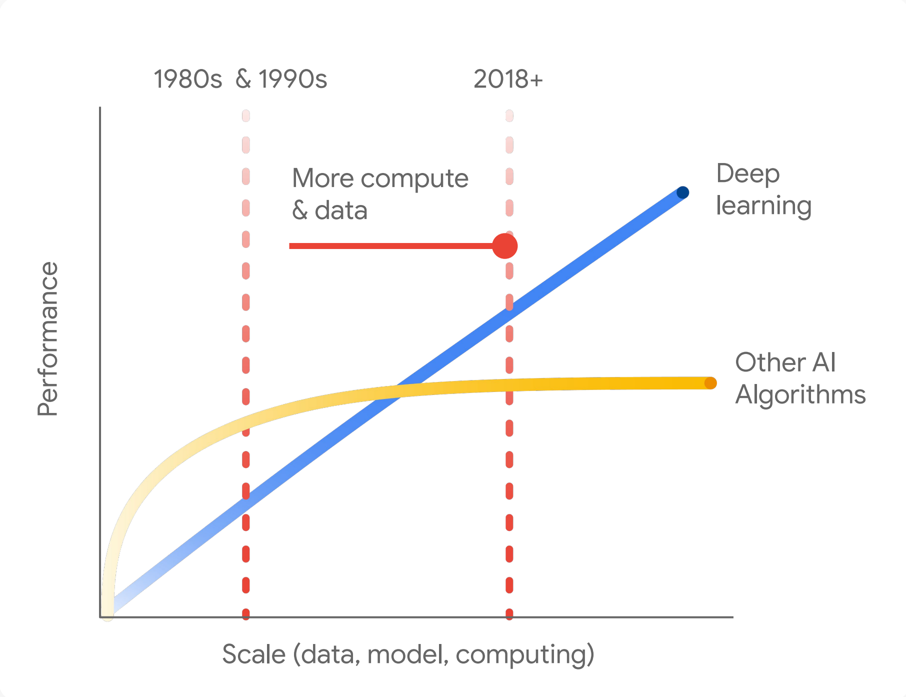
Direct attack point targeting



Efficient probabilistic attack

Better and intuitive success metrics

Attacks are going to be better over time as deep learning scales with data and computing

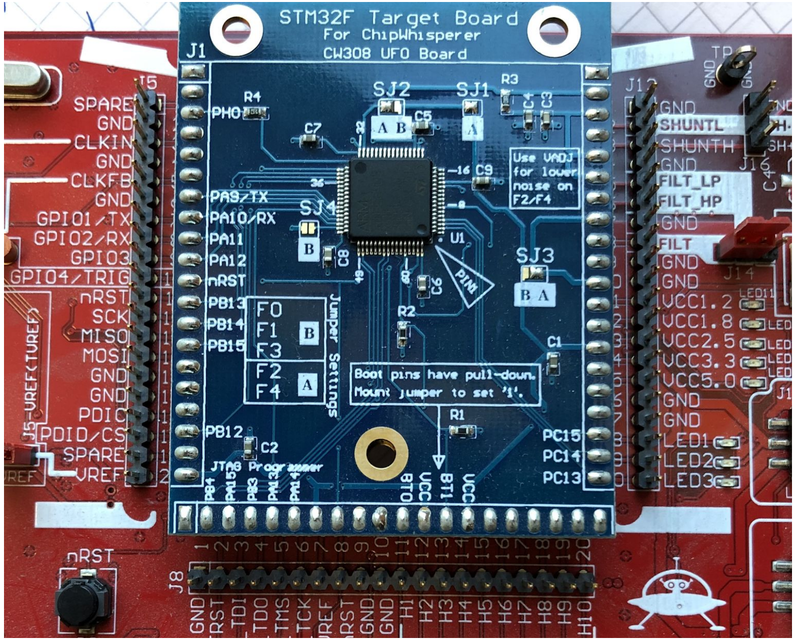


Talks, tutorial, code

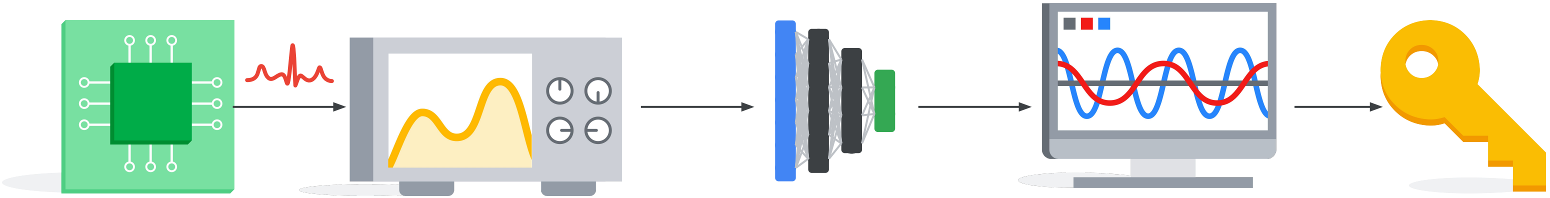
https://elie.net/scaaml

(and one day a paper)

Goal: train a model that can recover the AES keys from the STM32F415 TinyAES implementation using as few power traces as possible

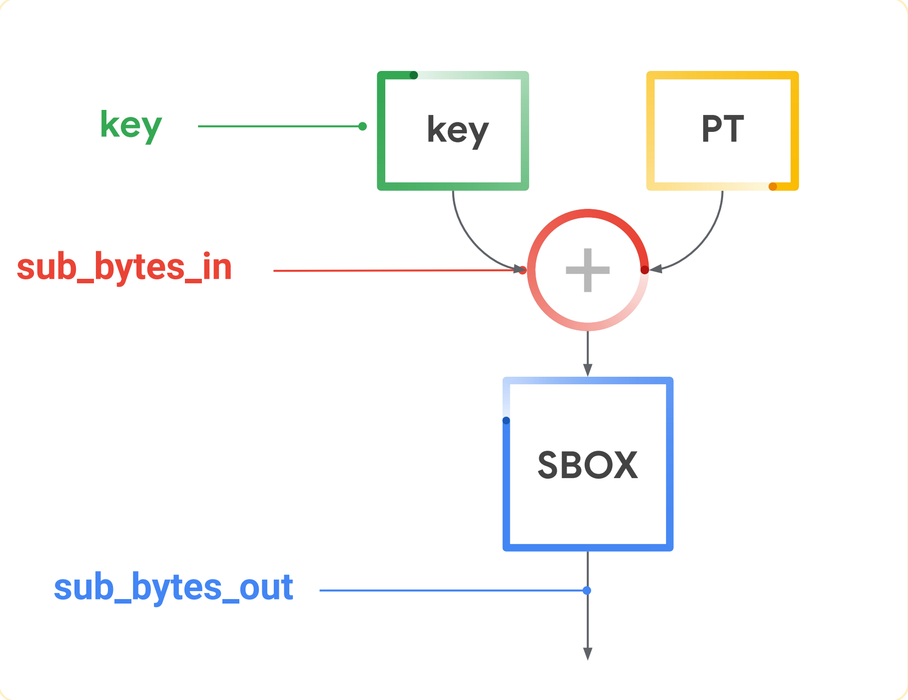


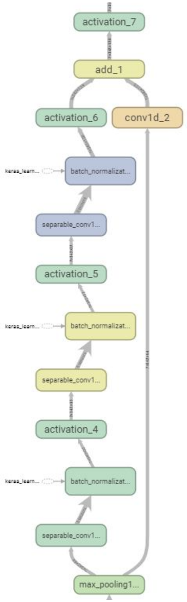
Game plan



Encryption → **Signal acquisition (cw + pico)** → **Predictions using DNN** → **Combine DNN predictions** → **AES key!**

TinyAES has multiple attack points that can be targeted by SCAAML.



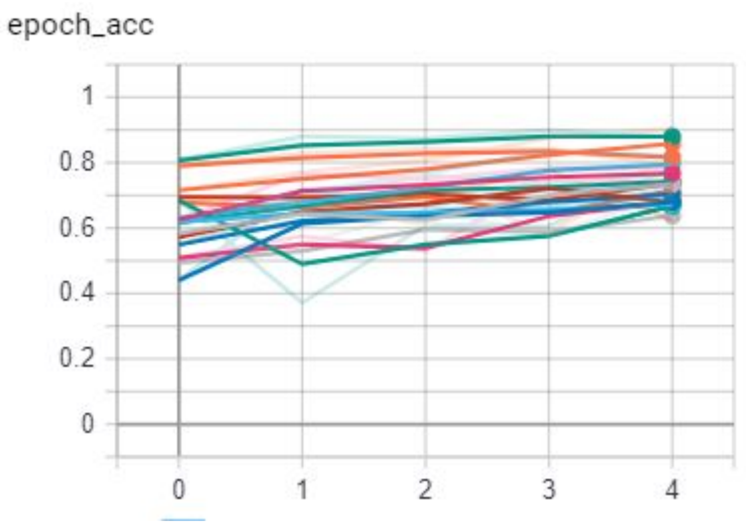


Custom residual block used

Model architecture

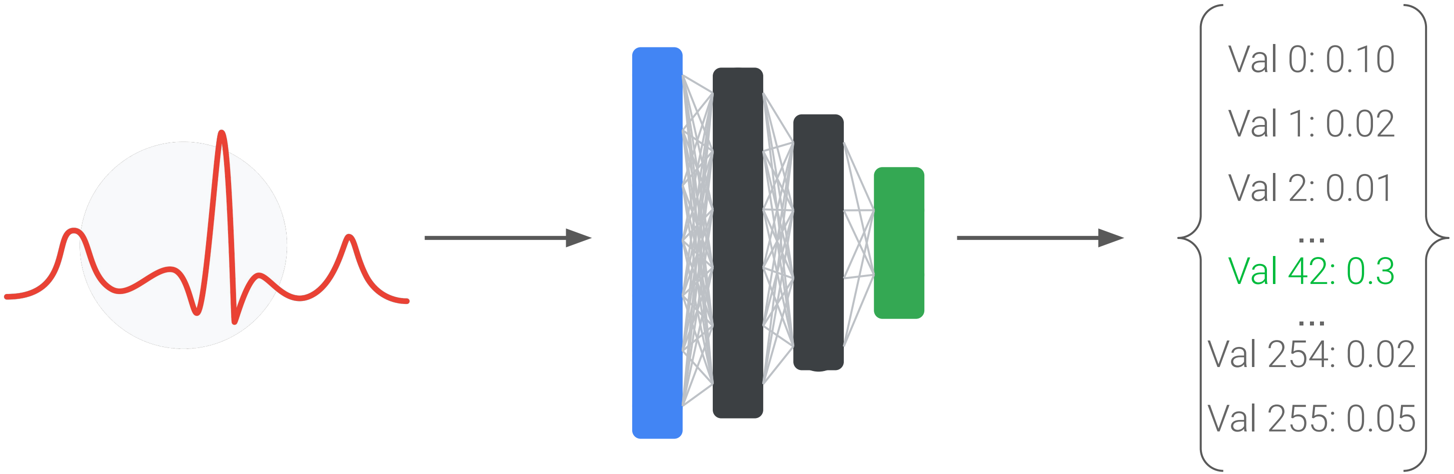
Hypertuned residual separated 1D convolution network

Tensorboards - 1 model per byte



Name	Smoothed	Value	Step
tinyaes_sync-cnn-v3-ap_sub_bytes_in-byte_0-len_8000\validation	0.8795	0.8787	4
tinyaes_sync-cnn-v3-ap_sub_bytes_in-byte_1-len_8000\validation	0.8165	0.7926	4
tinyaes_sync-cnn-v3-ap_sub_bytes_in-byte_10-len_8000\validation	0.7671	0.7822	4
tinyaes_sync-cnn-v3-ap_sub_bytes_in-byte_11-len_8000\validation	0.7345	0.7798	4
tinyaes_sync-cnn-v3-ap_sub_bytes_in-byte_12-len_8000\validation	0.6796	0.7205	4
tinyaes_sync-cnn-v3-ap_sub_bytes_in-byte_13-len_8000\validation	0.6722	0.6948	4
tinyaes_sync-cnn-v3-ap_sub_bytes_in-byte_14-len_8000\validation	0.6673	0.787	4
tinyaes_sync-cnn-v3-ap_sub_bytes_in-byte_15-len_8000\validation	0.8582	0.9032	4
tinyaes_sync-cnn-v3-ap_sub_bytes_in-byte_2-len_8000\validation	0.6791	0.6245	4
tinyaes_sync-cnn-v3-ap_sub_bytes_in-byte_3-len_8000\validation	0.6799	0.7369	4
tinyaes_sync-cnn-v3-ap_sub_bytes_in-byte_4-len_8000\validation	0.6377	0.702	4
tinyaes_sync-cnn-v3-ap_sub_bytes_in-byte_5-len_8000\validation	0.7029	0.7336	4
tinyaes_sync-cnn-v3-ap_sub_bytes_in-byte_6-len_8000\validation	0.7951	0.8205	4
tinyaes_sync-cnn-v3-ap_sub_bytes_in-byte_7-len_8000\validation	0.7423	0.7649	4
tinyaes_sync-cnn-v3-ap_sub_bytes_in-byte_8-len_8000\validation	0.7139	0.8047	4
tinyaes_sync-cnn-v3-ap_sub_bytes_in-byte_9-len_8000\validation	0.7366	0.803	4

Probabilistic attack: single trace



Probabilistic attack: summing predictions*

Val 0: 0.10
Val 1: 0.02
Val 2: 0.01
...
Val 42: 0.3
...
Val 254: 0.02
Val 255: 0.05

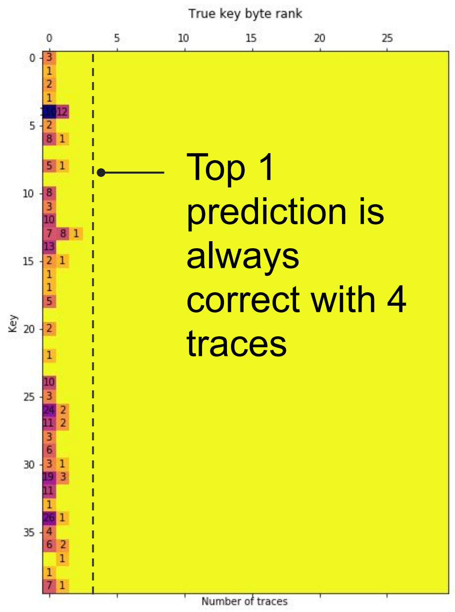
+ ... +

Val 0: 0.08
Val 1: 0.04
Val 2: 0.05
...
Val 42: 0.12
...
Val 254: 0.03
Val 255: 0.10

→

Val 0: 4.4
Val 1: 5.3
Val 2: 3.2
...
Val 42: 21.4
...
Val 254: 2.9
Val 255: 4.2

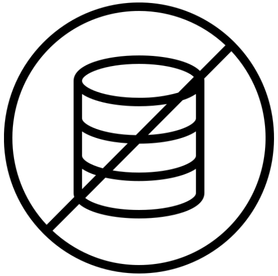
*in practice we sum uses log10 + ε to avoid numerical errors



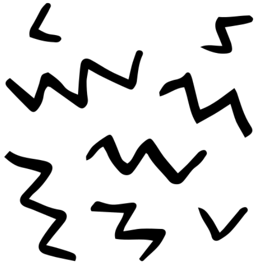
Despite having “only a 30% accuracy” our model allows to **recover automatically 100% of the bytes with at most 4 traces (81% with a single trace!) on a different chip**

Non suitable use-cases

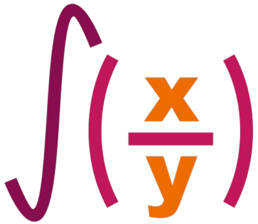
What makes a bad target?



Lack of data

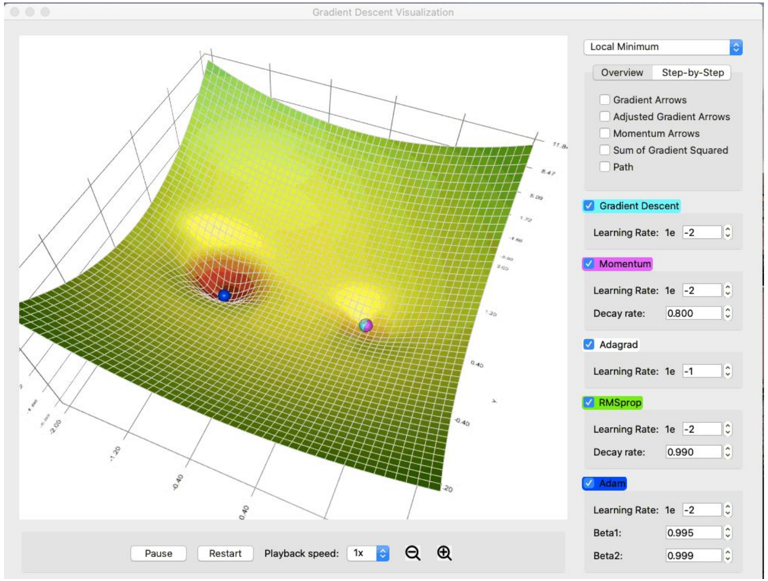


Lack of structure



Exact efficient solutions exist

Gradient descent needs a “smooth” domain to guide the descent



What NOT to do
use deep-learning to attack repeating ciphers

Google

Security and Privacy Group

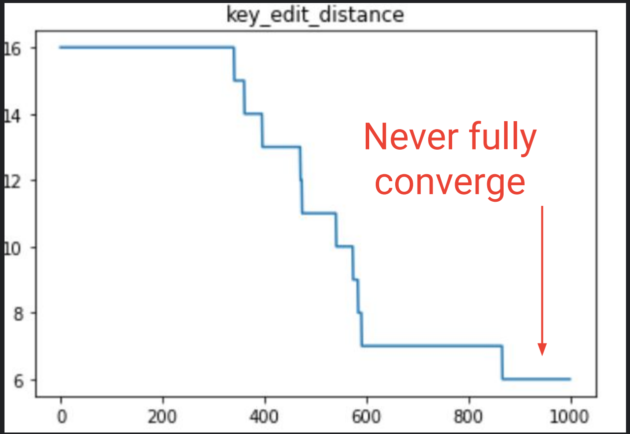
```python
    # decompress
    for _ in range(num_decompress_layers):
        m = D(m, num_dims, activation, use_batchnorm, dropout)
        num_dims = num_dims * 2

    # key layer
    key = layers.Dense(max_key_len, activation='sigmoid', name='key')(m)

    # plaintext
    plaintext = VignereDecrypt(ALPHABET_SIZE, max_key_len, output_dims, use_2d=use_2d,
                               name='plaintext', trainable=False)([key, cipher])

    return  Model(cipher, [plaintext, key])

model = make_model(BATCH_SIZE, input_dims, output_dims, MAX_KEY_LEN, dropout=0.2, activation='sig
model.summary()
```



Type	Predicition	Expected	L2 distance	Edit distance
key	ABCDBBADBACDBACD	ABCDABCDABCDABCD	3	6
pt	CRYPSOKSRIORSGOR	CRYPTOISSHORTFOR	3	6

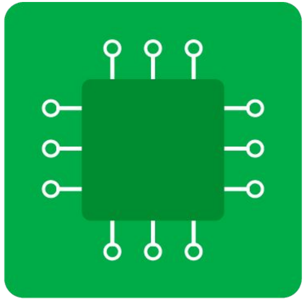
final loss: 0.0005950476097551059

If there isn't enough structure it will "mostly" work -- it's just not competitive

Unlocking new use-cases

That is the million dollar question

There is an underlying structure that can’t (easily) be explicitly expressed so we use deep-learning to approximate it



Maybe can recover keys from SOC that are too noisy for many SCA techniques

Self-supervised learning: learning without labels

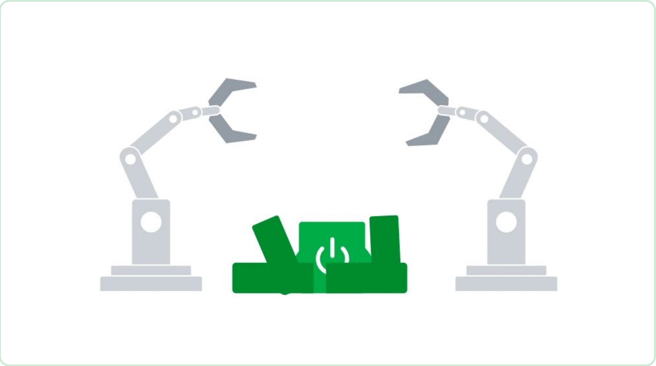
Can we adapt self-supervised technique to perform blackbox attacks?

Existing promising initial work

- **SCAUL: Power Side-Channel Analysis with Unsupervised Learning** Ramezanpour et al. 2020
- **Non-Profiled Deep Learning-based Side-Channel attacks with Sensitivity Analysis** Timon. 2019

Special tactics

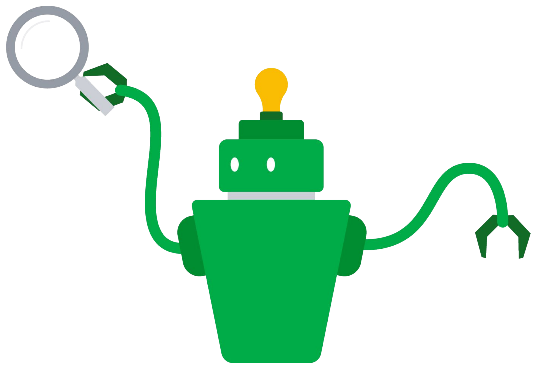
Repurpose deep-learning tooling for cryptanalysis



Google

Security and Privacy Group

Side Channel Attacks Leak Detector



Google

Security and Privacy Group

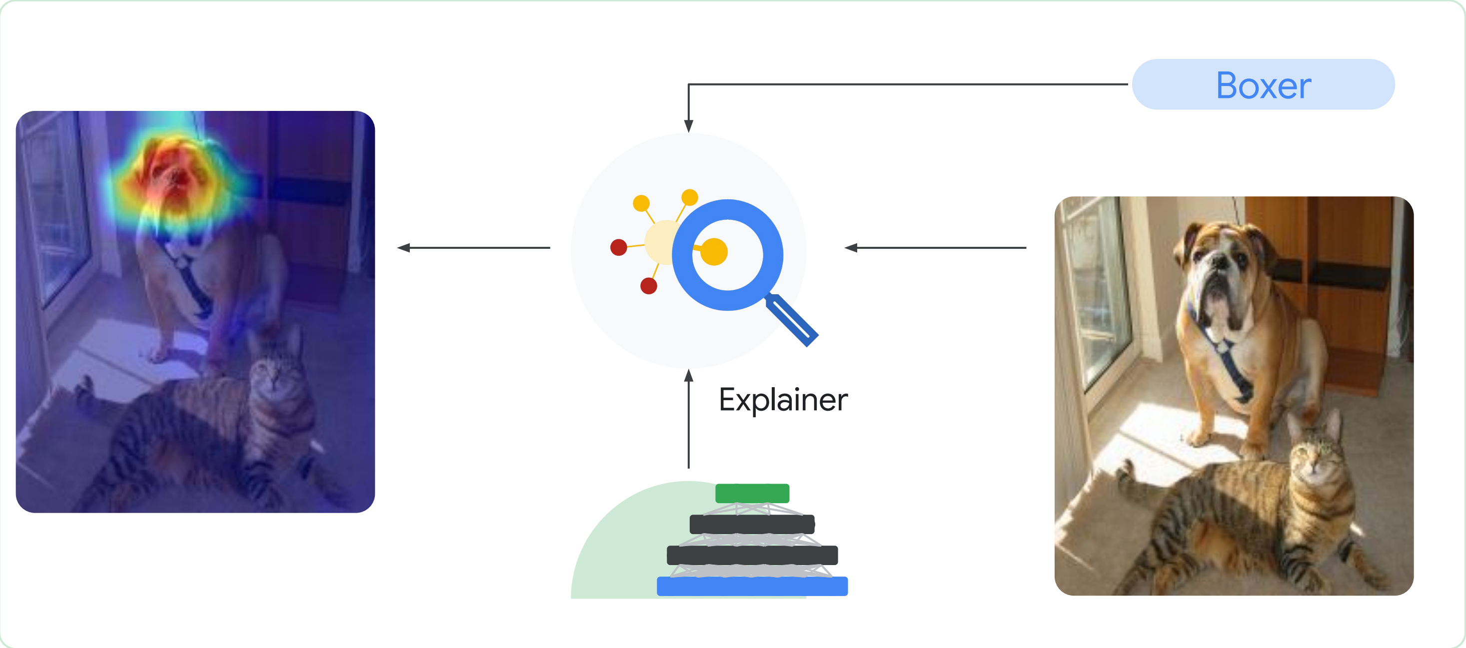
Slides + Video

https://elie.net/scald

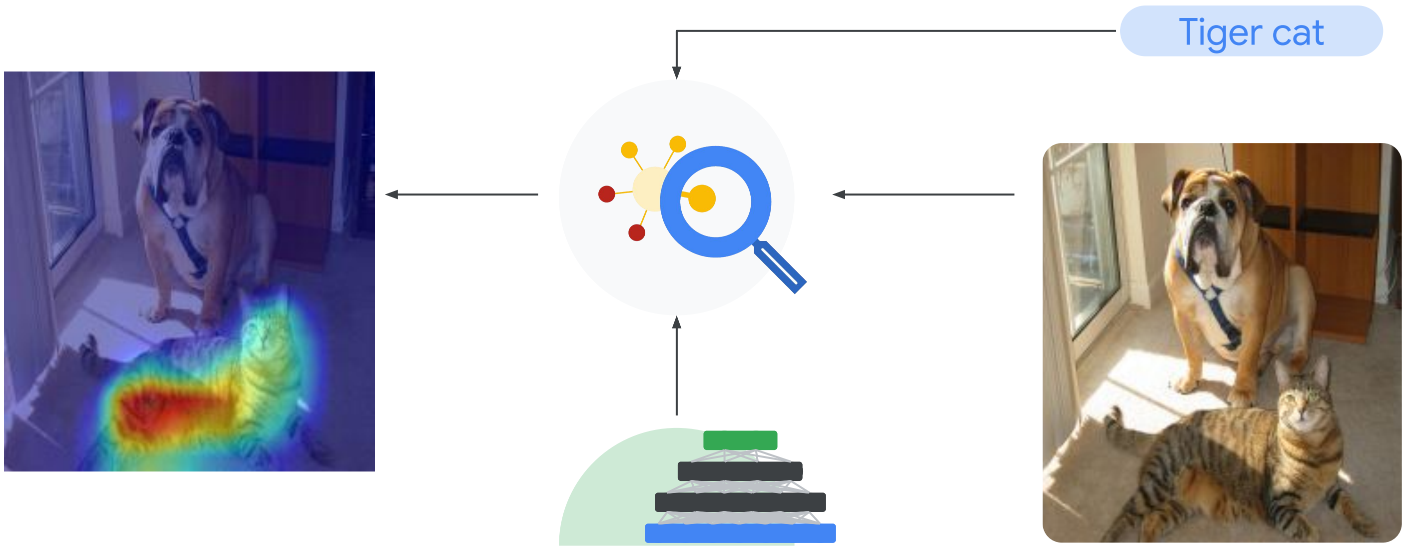
Google

Security and Privacy Group

Explainability techniques: boxer prediction

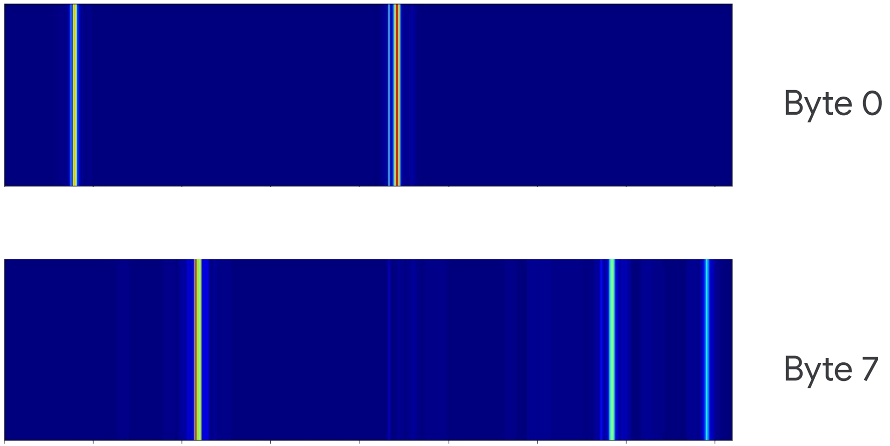


Explainability techniques: **cat prediction**

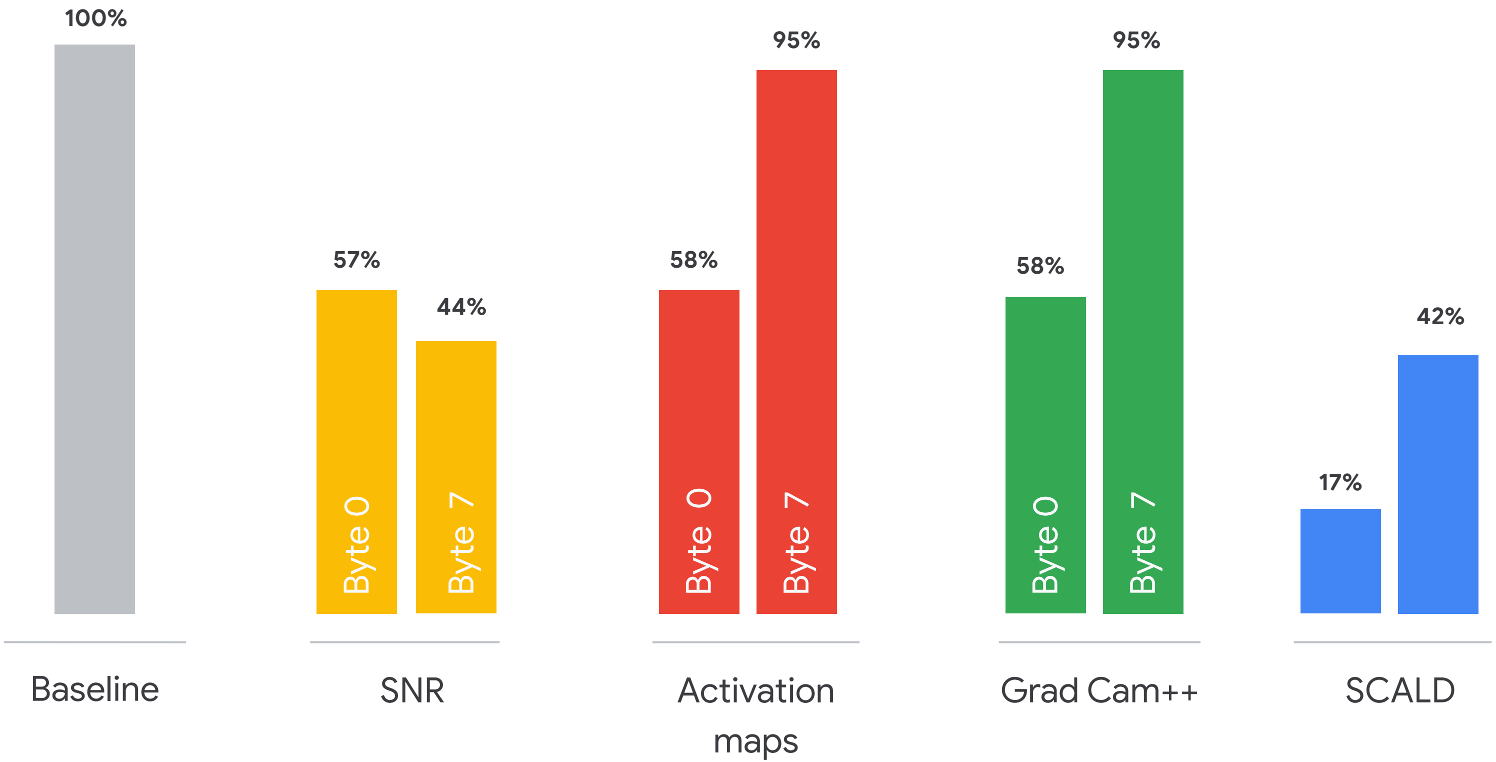


SCALD explainer combines partitioned and convolutive occlusion for speed and precise leakage pinpointing

SCALD leakage map



Benchmark results: lower is better



Google

Preliminary results - 4 points masked

Security and Privacy Group

```
stm32f415-tinyaes_sync
├──
│   ├── AES128_ECB_indp_crypto()
│   ├── AddRoundKey()
│   │   └── 0 - residual leakage (leak score:80)
│   ├── Cipher()
│   ├── ShiftRows()
│   ├── SubBytes()
│   ├── aes_indep_enc()
│   └── xtime()
├── aes.c
│   ├── AES128_ECB_indp_crypto()
│   ├── AddRoundKey()
│   │   ├── 207 - residual leakage (leak score:112)
│   │   └── 213 - Main leakage (leak score:240)
│   ├── Cipher()
│   │   ├── 276 - potential leakage (leak score:144)
│   │   ├── 277 - residual leakage (leak score:96)
│   │   ├── 278 - residual leakage (leak score:112)
│   │   ├── 279 - residual leakage (leak score:112)
│   │   ├── 280 - potential leakage (leak score:128)
│   │   ├── 371 - Secondary Leakage (leak score:176)
│   │   ├── 380 - residual leakage (leak score:96)
│   │   ├── 383 - residual leakage (leak score:112)
│   │   └── 393 - Secondary Leakage (leak score:176)
│   ├── ShiftRows()
│   │   └── 240 - residual leakage (leak score:96)
│   ├── SubBytes()
│   │   ├── 130 - potential leakage (leak score:128)
│   │   ├── 221 - residual leakage (leak score:80)
│   │   └── 227 - residual leakage (leak score:80)
│   └── xtime()
│       └── 265 - residual leakage (leak score:80)
├── simpleserial-aes.c
│   └── get_pt()
└── stm32f4_hal_lowlevel.c
    └── HAL_GPIO_WritePin()
```

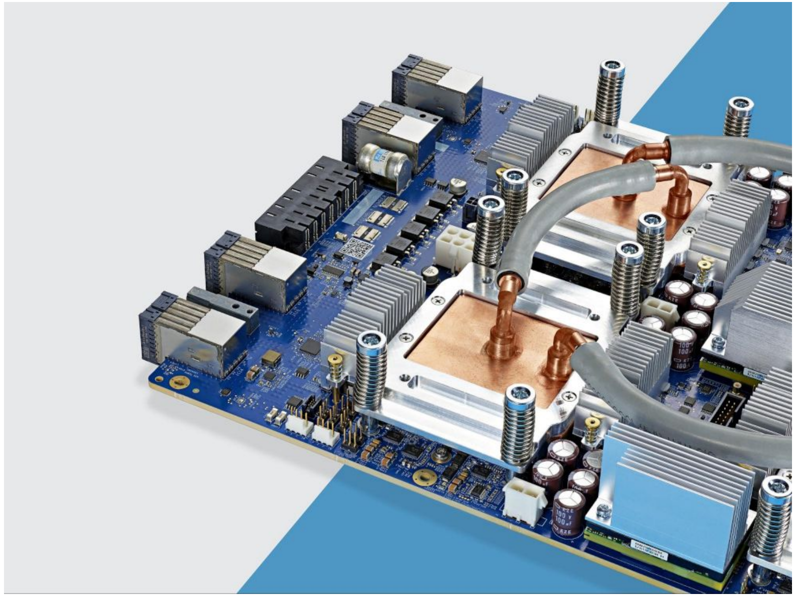
SCALD analysis result output

aes.c ×

scald > firmwares > tinyaes_src > aes.c > AddRoundKey(uint8_t)

```c
203
204 // This function adds the round key to state.
205 // The round key is added to the state by an XOR function.
206 static void AddRoundKey(uint8_t round)
207 {
208   uint8_t i,j;
209   for(i=0;i<4;++i)
210   {
211     for(j = 0; j < 4; ++j)
212     {
213       (*state)[i][j] ^= RoundKey[round * Nb * 4 + i * Nb + j];
214     }
215   }
216 }
217
```

TinyAES aes.c line 213 is **exactly** the sub_byte_in operation! SCALD perfectly identifies the main source of leakage.



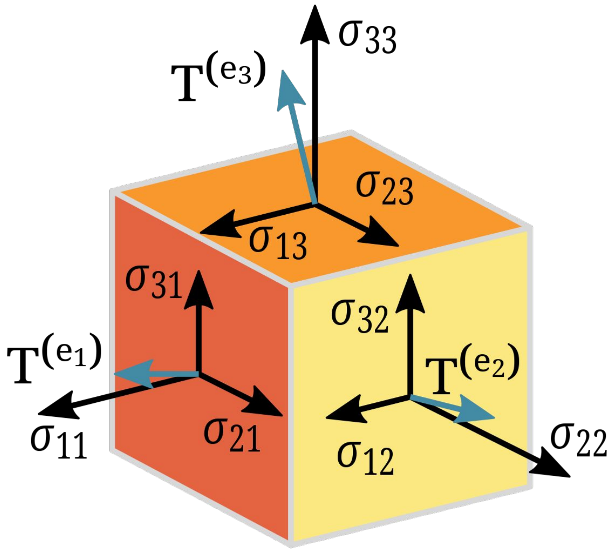
Leverage deep-learning hardware for cryptanalysis

Write cryptanalysis algorithm as tensor operations

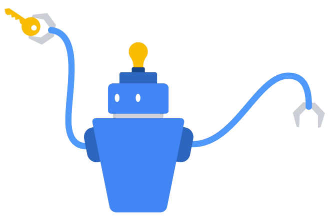
Benefit from well designed high-level libraries such as Jax, and TensorFlow

Get highly optimized hardware accelerated distributed code

Short path vector anyone?

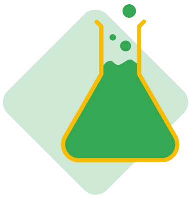


Takeaways



Deep-learning is a natural fit for SCA (SCAAML)

Making implementations that are deep-learning resilience is becoming critical



AI for cryptanalysis is still a nascent field with a lot of exciting opportunities

Applying deep-learning to cryptanalysis is an exciting new area full of promises. Join the revolution:)