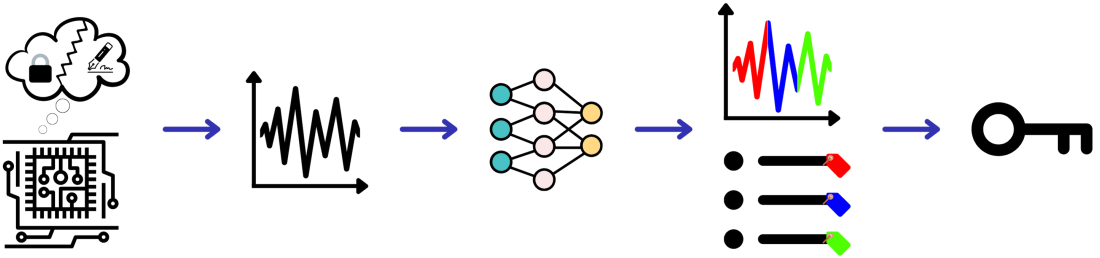
Neural network attacks using side-channel leakages to break ECC implementations on consumer electronics

Michele Corrias - 808746
Supervisor: Prof. Danilo Bruschi Co-supervisor: Eng. Guido Bertoni Examiner: Prof. Vincenzo Piuri

University of Milan
Security Pattern

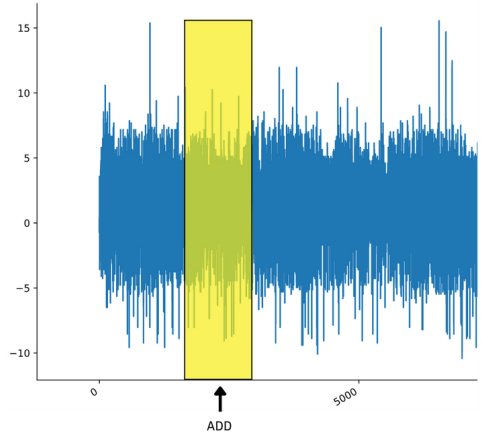
June 09, 2022

Breaking The Key



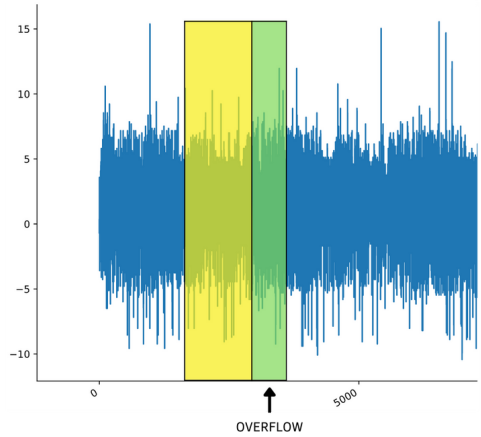
Background

```
ADD(x, y, modulo)
begin
    result = x + y
    if result ≥ modulo then
        return result - modulo
    else
        return result
end
```



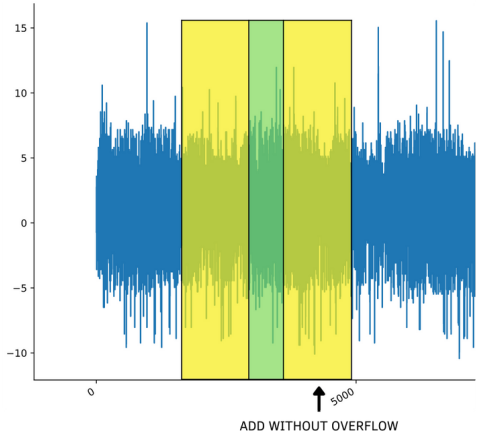
Background

```
ADD(x, y, modulo)
begin
    result = x + y
    if result ≥ modulo then
        return result - modulo
    else
        return result
end
```

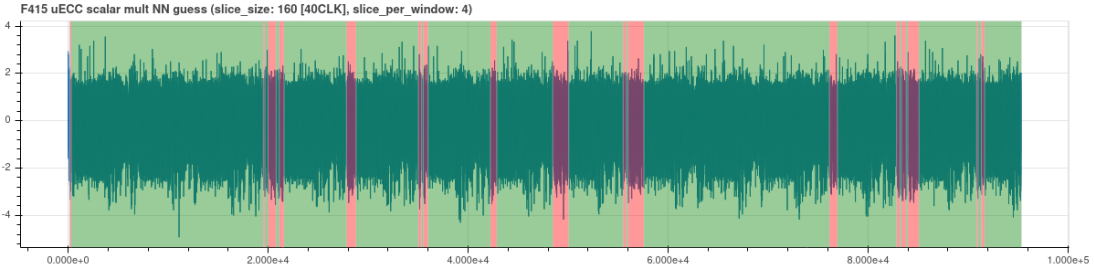


Background

```
ADD(x, y, modulo)
begin
    result = x + y
    if result ≥ modulo then
        return result - modulo
    else
        return result
end
```



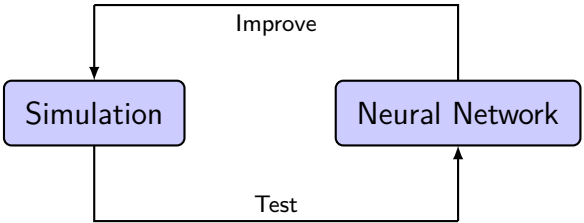
Previous Results

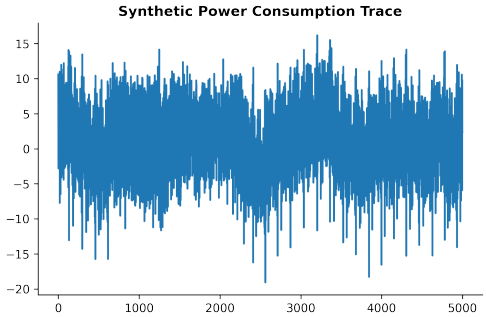


Accuracy on a single trace of the previous work

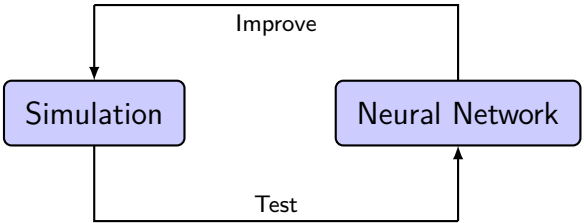
- *correct predictions* and *wrong predictions*
- $\text{accuracy} = \dfrac{\text{correct classifications}}{\text{total windows of samples}}$

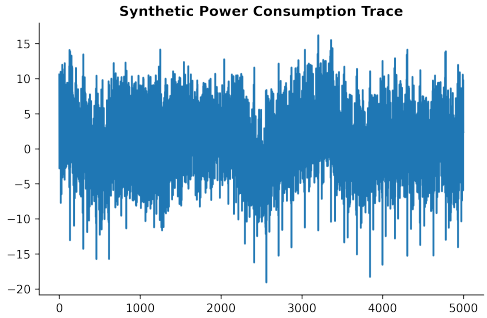
Simulation Environment





Simulation Environment





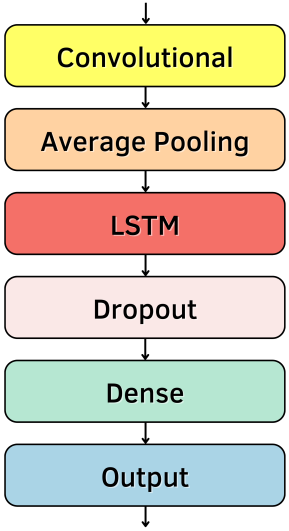
Improvements

- dataset balancing and scaling
- horizontal component
- vertical and horizontal noises

Simulation Results

- synthetic trace $\sim$ real trace
- STOP with 100% accuracy

LSTM



Input

Dataset from a power consumption trace of a cryptographic computation

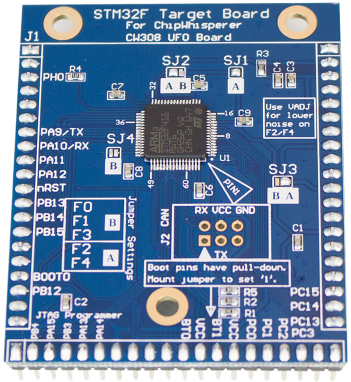
LSTM

- sequences of data and time series
- memory over time and temporal consciousness
- *Human Activity Recognition*

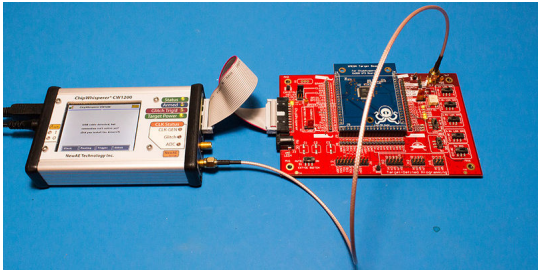
Output

Binary classification of the windows composing the dataset

Experiment



(a) STM32f415RG

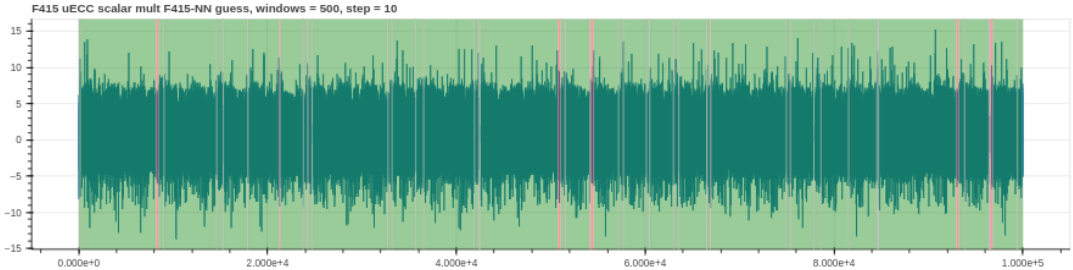


(b) ChipWhisperer-Pro

A few numbers

- trace size $\in [7, 10]$ million samples
- dataset size $\simeq$ 700 thousand windows
- window size $=$ 500 samples

Results



Neural Network Properties

- LSTM *correct predictions* and *wrong predictions*
- accuracy $= \dfrac{\text{correct classifications}}{\text{total windows of samples}} > \mathbf{97\%}$
- LSTM able to recognize the length of modular operations

Conclusions & Future Works

Conclusions

- success in automatic recognition of operations during a ECC computation
- LSTM accuracy allows a key-recovery attack $\rightarrow$ an attacker could retrieve the entire user private key

Work highlights:

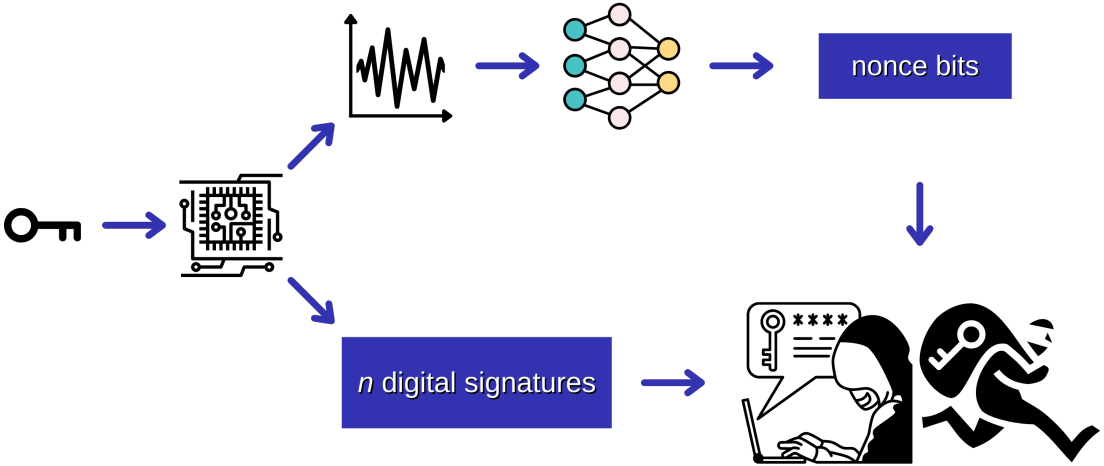
- focus on MCU running ECC implementation
- SCA on power consumption
- NN implementation, training and testing

Future Works

- LSTM testing with different SCAs on different microcontrollers and software libraries
- paper

Thanks for your attention!

Key Recovery



Datasets Sizes

Table: Sizes of four real power consumption traces, acquired for testing, with derived datasets

Trace	*Trace Size*	*Trace Samples*		*Dataset size*	*Dataset Windows*	
		SO samples	*LO samples*		`true`-*tag*	`false`-*tag*
`T1`	7032251	524697	6507554	703176	86948	616228
`T2`	7002209	522537	6479672	700171	86739	613432
`T3`	7005521	525957	6479564	700503	87086	613417
`T4`	7027186	524247	6502939	702669	86893	615776

Accuracy Results

Table: Accuracy results on four real power consumption traces

Trace	*Predictions*		*Accuracy*	*Loss*
	#right	*#wrong*		
T1	686165	17011	0.9758	0.0621
T2	683800	16371	0.9766	0.0603
T3	683410	17093	0.9756	0.0646
T4	683536	19133	0.9728	0.0702

Similarity Metrics

Table: Results of similarity metrics to compare raw predictions with raw ground-truth

ID	*VI*	*NMI*	*AMI*	*ARS*	*VM*
T1	0.0	0.756	0.756	0.867	0.756
T2	0.0	0.757	0.757	0.871	0.757
T3	0.0	0.752	0.752	0.866	0.752
T4	0.0	0.738	0.738	0.852	0.738

Post-processing Results

Table: Post-processing results on the four real power consumption traces

Trace	*Post-processed Ground-truth* (total #SO)	*Post-proc. Ground-truth SO performing modulus* #yes	#no	*Post-processed Predictions* (total #SO)	*Post-proc. Predictions SO performing modulus* #yes	#no
T1	1770	1209	561	1803	1386	417
T2	1770	1188	582	1799	1323	476
T3	1770	1223	547	1803	1376	427
T4	1770	1214	556	1794	1440	354

Scalar Multiplication & ECDLP

Scalar Multiplication

k integer, P point of $\mathcal{E}_{\mathbb{F}_p}$ elliptic curve: the scalar multiplication kP is the result of adding P to itself k times

$$kP = \underbrace{P + P + \cdots + P}_{k \text{ TIMES}} \tag{1}$$

The scalar multiplication is the core operation of ECC, in which the scalar is usually secret.

Elliptic Curve Discrete Logarithm Problem

Elliptic curve $\mathcal{E}_{\mathbb{F}_p}$ defined over a finite field $\mathbb{F}_p$, two points G and Q of the curve.
ECDLP: find a positive integer k, if there exists, such that:

$$Q = kG \tag{2}$$

where kG is the result of scalar multiplication between k and G.

Differential Power Analysis

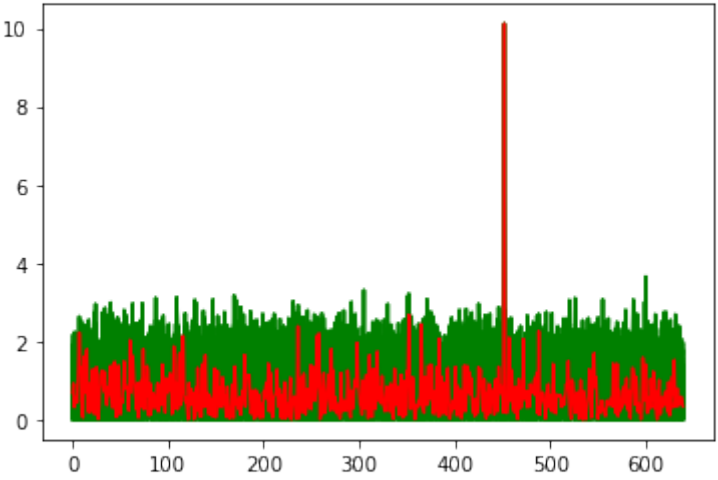


Figure: key $= 2^{160} - 9839030998049910$ - n $= 250$ - 8 bits to guess

ECDSA Sign

ECDSA

- authentication
- non-repudiation

---

procedure ECDSA SIGN
input: d private key, m message
output: (r, s) signature of m
    k random integer $\in [1, n-1]$
    $(x_1, y_1) = k \times G \mid G$ base point, $n \times G = \mathcal{O}$
    $r = x_1 \mod n$
    $e_{10} = \text{HASH}(m)$
    $s = k^{-1}(e + dr) \mod n$
    return: (r, s)
end procedure

ECDSA Verify

procedure ECDSA VERIFY
input: r,s, m message
output: (r, s) signature of m
    (r, s) $\in [1, n-1]$
    e = HASH (m)
    $u_1 = es^{-1} \mod n$
    $u_2 = rs^{-1} \mod n$
    Calculate the curve point $(x_1,y_1) = u_1 \times G + u_2 \times Q_A$
    if $(x_1,y_1) = \mathcal{O}$ then signature invalid
    if $r \equiv x_1 \pmod n$ then signature valid
    else **return** signature invalid
end procedure

Power Consumption as Leakage

- realization of power analysis is easier than anything else
- quality of signal
- number of traces to collect

Conditions

- in `secp160r1` nonce k is random scalar 160 bits long
- in `EccPoint_mult` scalar multiplication:
  1. `XYCZ-IDBL`
  2. a cycle of 160 iterations on 160 bits of nonce k, starting from MSB
     - `XYcZ_addC` e `XYcZ_add` in the cycle
  3. the last bit of nonce k, at LSB, outside of the cycle
- *affines* standard coordinates in two dimensions
- projective coordinates: different possible representations of a point P on an elliptic curve, as by a triplet (X, Y, Z)
- a point has several projective coordinates, as many as different Z
- Montgomery ladder with (X, Y)-only co-Z addition implements scalar multiplication
  - *projective* coordinates: faster
- first, `initial_Z` $= 1$ in `XYCZ-IDBL`
- then, no restrictions on `initial_Z` in the paper

Fingerprint Operations

Montgomery ladder with (X, Y)-only co-Z addition:

- b current target bit
- $(R_{1-b}, R_b) \leftarrow \texttt{XYCZ-ADDC}(R_b, R_{1-b}) = \text{P} + \text{Q}, \text{P - Q}$
- $(R_b, R_{1-b}) \leftarrow \texttt{XYCZ-ADD}(R_{1-b}, R_b) = \text{P} + \text{Q}, \text{P}$
- fixed P, Q $\in \mathcal{E}$, the operands depend directly on the value of the processed bit of the scalar

For the first bit of the nonce k:

- according to Montgomery ladder, $R_0 = \text{P}$, $R_1 = 2\text{P}$

if b = 0

$(R_1, R_0) \leftarrow \texttt{XYCZ-ADDC}(R_0, R_1) = \texttt{XYCZ-ADDC}(P, 2P) = (3P, -P)$
$(R_0, R_1) \leftarrow \texttt{XYCZ-ADDC}(R_1, R_0) = \texttt{XYCZ-ADD}(3P, -P) = (2P, 3P)$

if b = 1

$(R_0, R_1) \leftarrow \texttt{XYCZ-ADDC}(R_1, R_0) = \texttt{XYCZ-ADDC}(2P, P) = (3P, P)$
$(R_1, R_0) \leftarrow \texttt{XYCZ-ADDC}(R_0, R_1) = \texttt{XYCZ-ADD}(3P, P) = (4P, 3P)$

GPUs

Cloud Service	NVIDIA GPU	CUDA Version	GPU RAM (GB)	CPU Chip	Chip Speed (GHz)	CPU Cores	CPU RAM (Total GB)	L3 Cache (MB)	Disk Space (Total GB)	Max Idle Time (hrs)	Max Session Time (hrs)	Max Commit Time (hrs)
Colab	Tesla K80	10.0	12.0	Intel Xeon CPU	2.2 or 2.3	2	13.3	56	359	1.5	12	n/a
Kaggle	Tesla P100	9.2	17.1	Intel Xeon CPU	2.2 or 2.3	2	16.4	46	220	1	9	6
Command	!nvidia-smi	!cat /usr/local/cuda/version.txt	!nvidia-smi	!cat /proc/cpuinfo	!cat /proc/cpuinfo	multiprocessing.cpu_count()	!cat /proc/meminfo	!cat /proc/cpuinfo	!df -h	docs	docs	docs