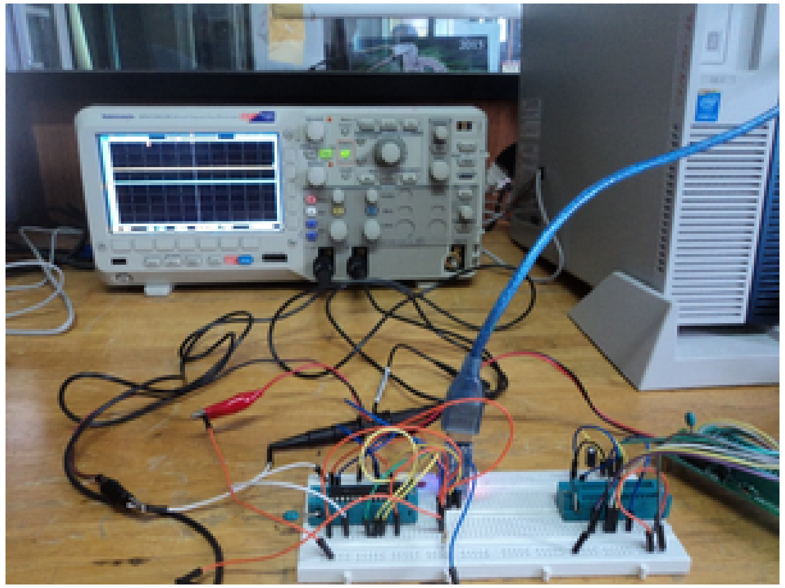
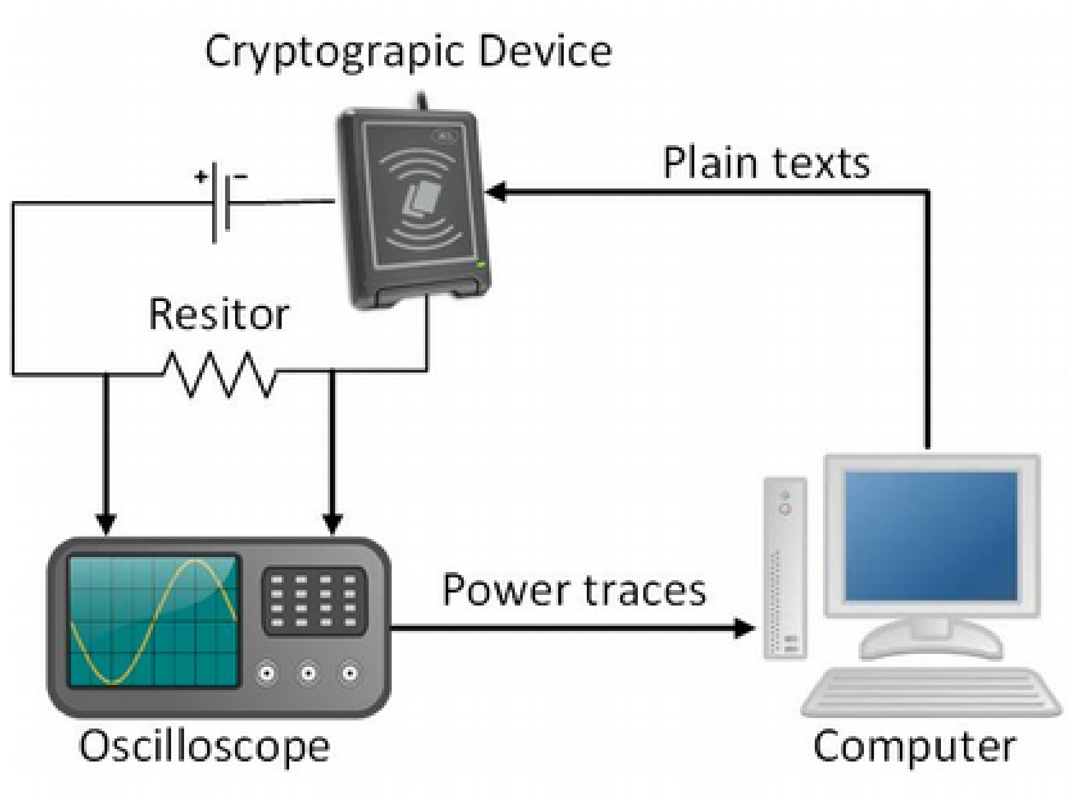
POWER ANALYSIS ATTACK



Group Members	Supervisors
H.M.GAMAARACHCHI (E/10/102)	DR. ROSHAN RAGEL
P.B.H.B.B.GANEGODA (E/10/104)	DARSHANA JAYASINGHE (PHD STUDENT/UNSW)

Power analysis attack



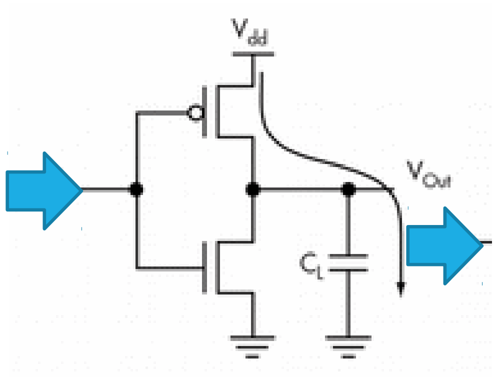
- Break the key of the system by measuring power consumption of the cryptosystem
- First collect power traces
- Then do statistical analysis using Pearson Correlation to get the secret key
  - Simple power analysis (SPA)
  - Differential power analysis (DPA)

Target devices for the attack

- Security of embedded devices such as smartcards was completely shattered when power analysis attack was introduced
- But today various countermeasures have been applie

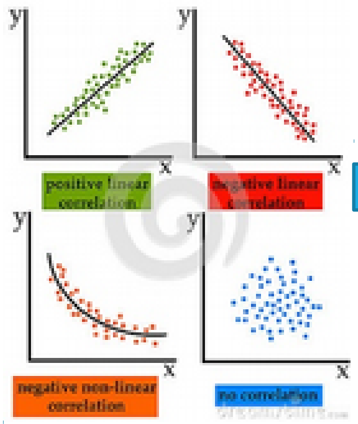
Technical approach

Secret key



Dynamic power consumption of CMOS circuits

Measured power traces



Pearson correlation

Secret key

key guesses

Power model
Eg : Hamming weight

$$\hat{\rho} = \frac{N\sum_{i=0}^{N} W_{i,j}H_i - \sum_{i=0}^{N} W_{i,j}\sum_{i=0}^{N} H_i}{\sqrt{N\sum_{i=0}^{N} W_{i,j}^2 - \left(\sum_{i=0}^{N} W_{i,j}\right)^2}\sqrt{N\sum_{i=0}^{N} H_i^2 - \left(\sum_{i=0}^{N} H_i\right)^2}}$$

$$\hat{\rho} = \frac{N\sum_{i=0}^{N} W_{i,j} H_i - \sum_{i=0}^{N} W}{\sqrt{N\sum_{i=0}^{N} W_{i,j}^2 - ¿}}$$

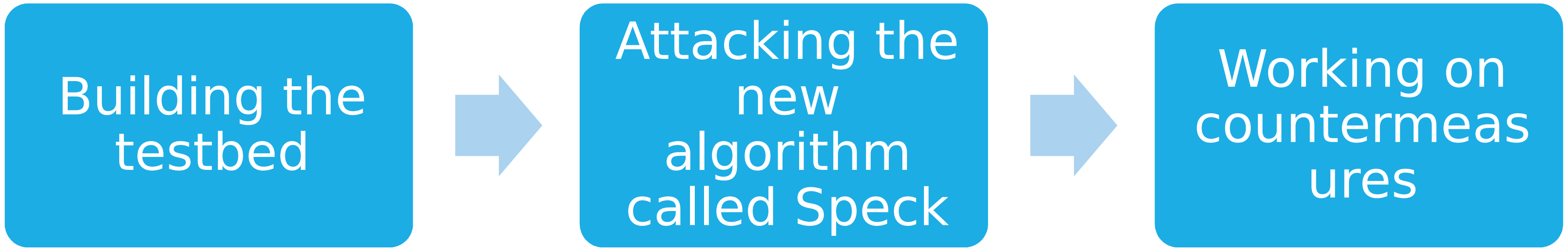
Why we do it?

- Why there is cryptography on the first place?
- What if a cryptographic system is insecure?

Therefore, attack with a genuine intention to help finding vulnerabilities and countermeasures.

Also there is a bit of fun when attacking a crypto sy

Project phases

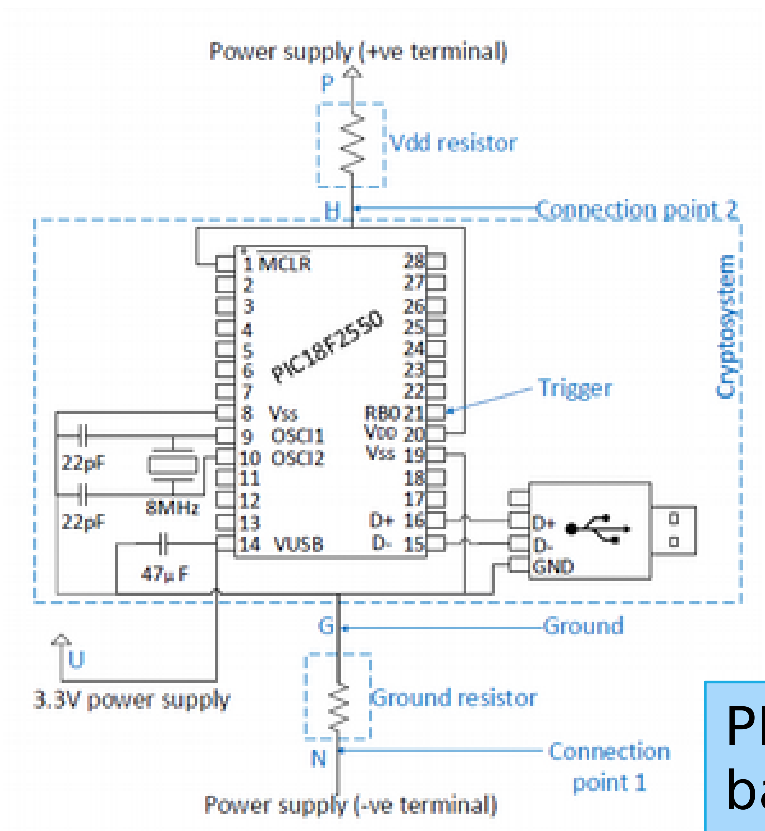

Building the testbed
Attacking the new algorithm called Speck
Working on countermeasures

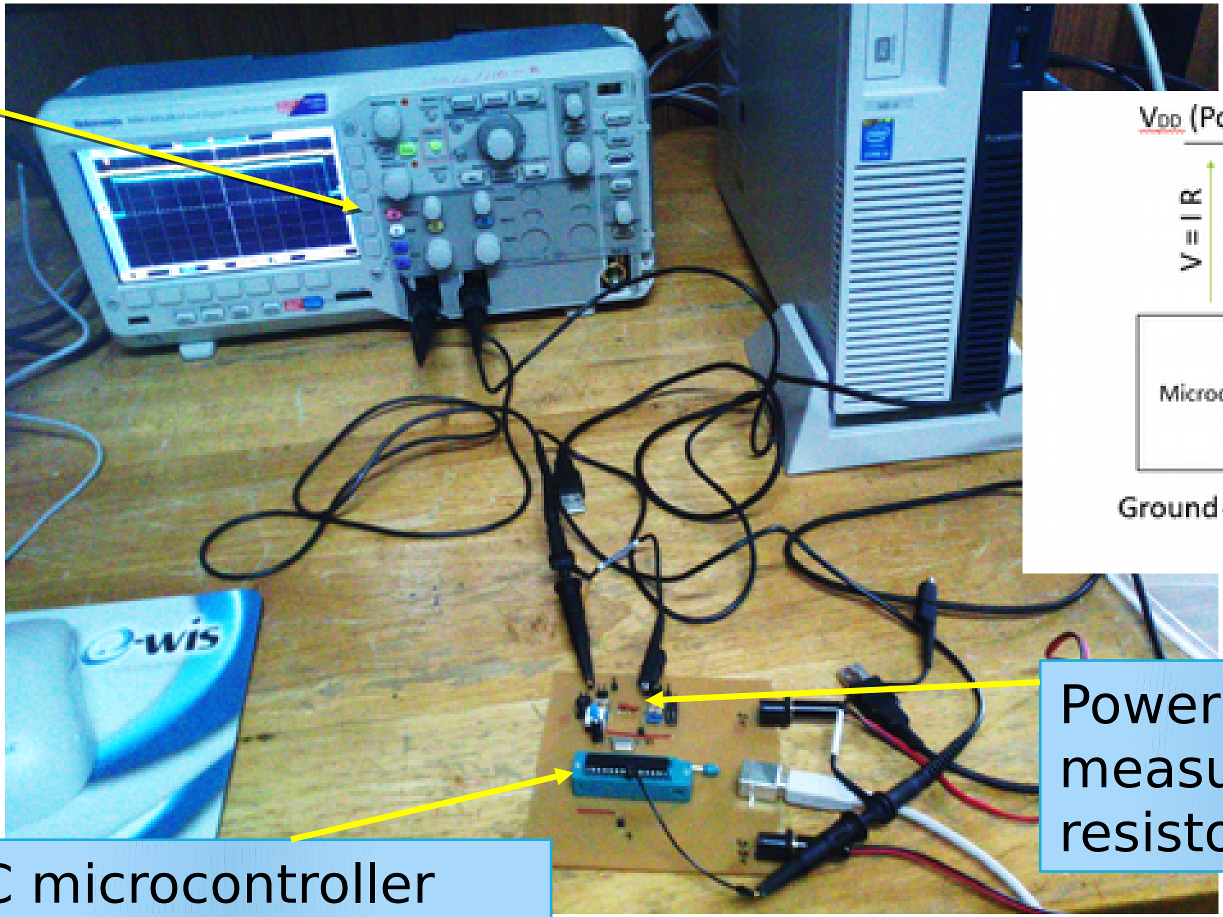
Phase 1

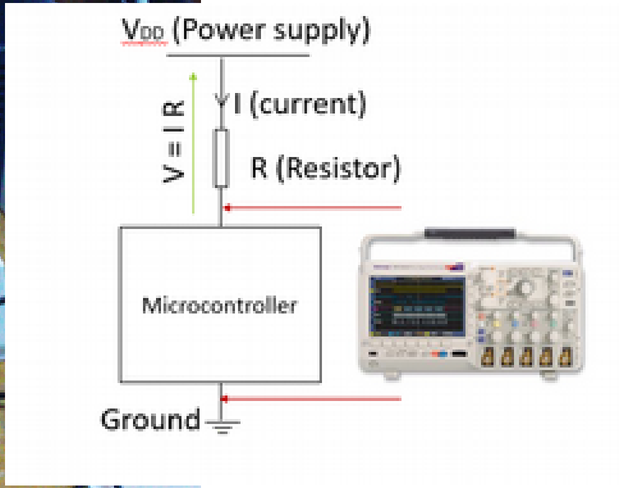
Building the testbed

Testbed : Hardware part

Digital Oscilloscope







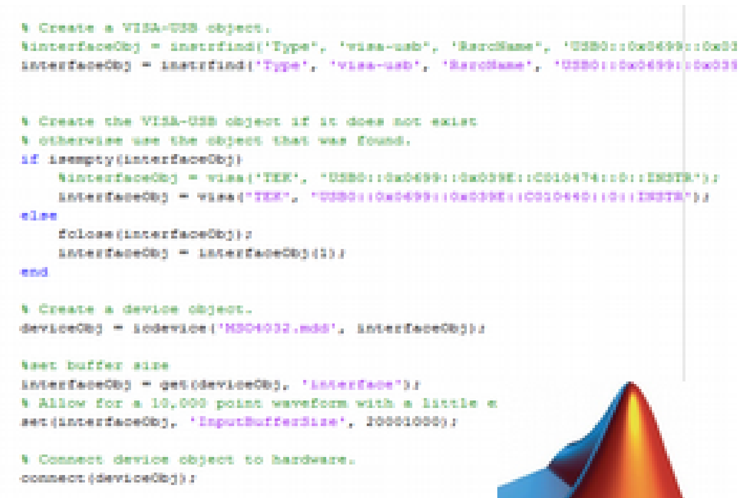
Power measuring resistor

PIC microcontroller based cryptographic device

Testbed : Software

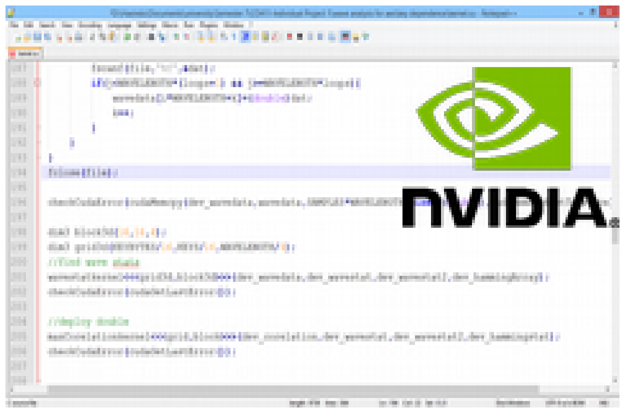
```
#include <18F2550.h>
[illegible]
#define key_size 128
[illegible]
```

Encryption Program on PIC



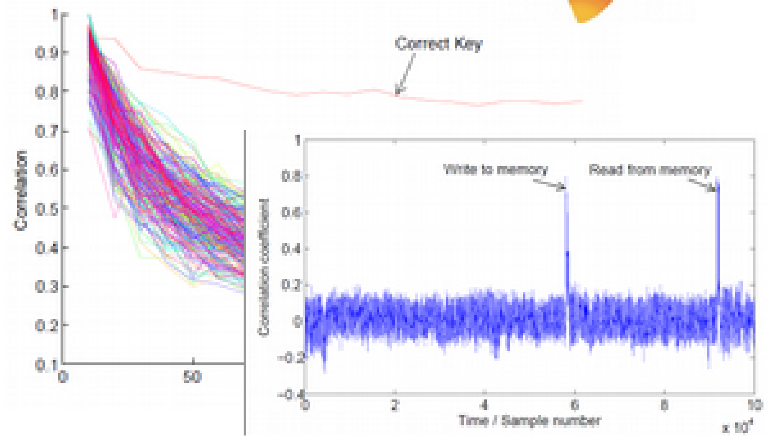
```
% Create a VISA-USB object.
%interfaceObj = instrfind('Type', 'visa-usb', 'RsrcName', 'USB0::0x0699::0x03
interfaceObj = instrfind('Type', 'visa-usb', 'RsrcName', 'USB0::0x0699::0x03

% Create the VISA-USB object if it does not exist
% otherwise use the object that was found.
if isempty(interfaceObj)
    %interfaceObj = visa('TEK', 'USB0::0x0699::0x039E::C010474::0::INSTR');
    interfaceObj = visa('TEK', 'USB0::0x0699::0x039E::C010440::0::INSTR');
else
    fclose(interfaceObj);
    interfaceObj = interfaceObj(1);
end

% Create a device object.
deviceObj = icdevice('MSO4032.mdd', interfaceObj);

%set buffer size
interfaceObj = get(deviceObj, 'interface');
% Allow for a 10,000 point waveform with a little e
set(interfaceObj, 'InputBufferSize', 20001000);

% Connect device object to hardware.
connect(deviceObj);
```

Instrument Control for automation



```
[illegible]
fclose(file);
[illegible]
//find more stats
[illegible]
//deploy double
[illegible]
```

CPA on CUDA



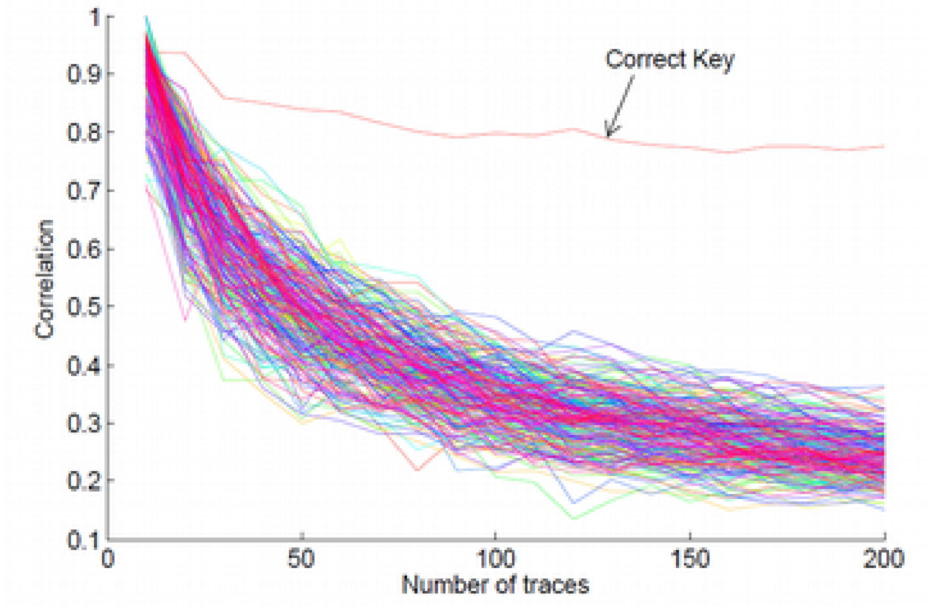
Scripts for post analysis

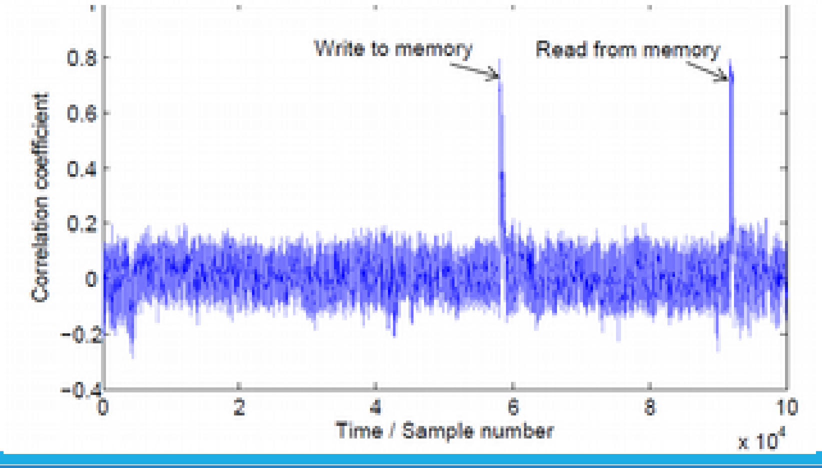
Breaking AES on the testbed

Step	Time taken / s
Collecting power traces	370
Running CPA	8
Sum	**378**

No of power traces required : 200

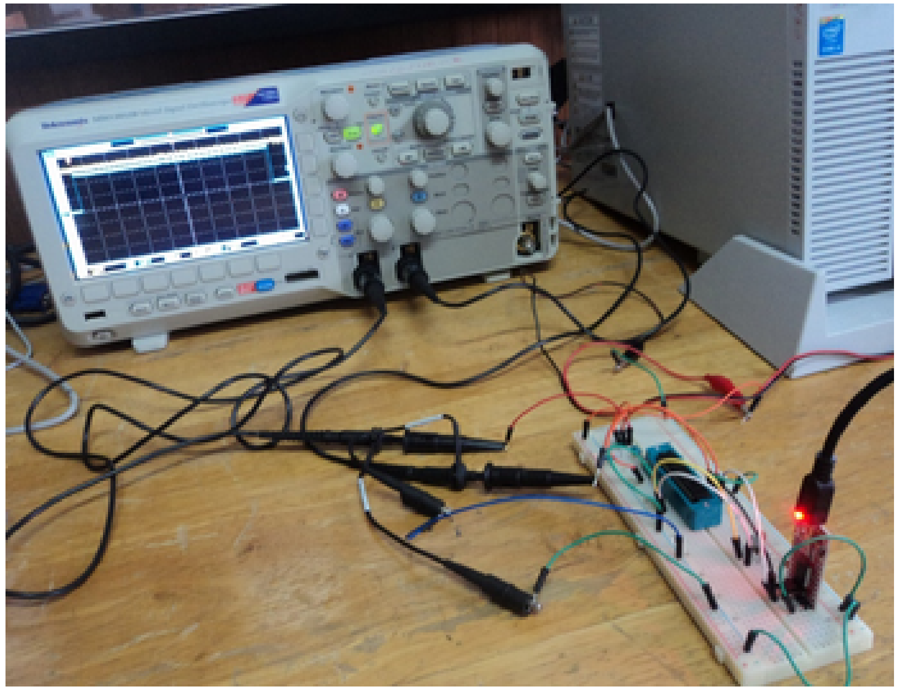
Time : Less than 10 minutes





Contributions

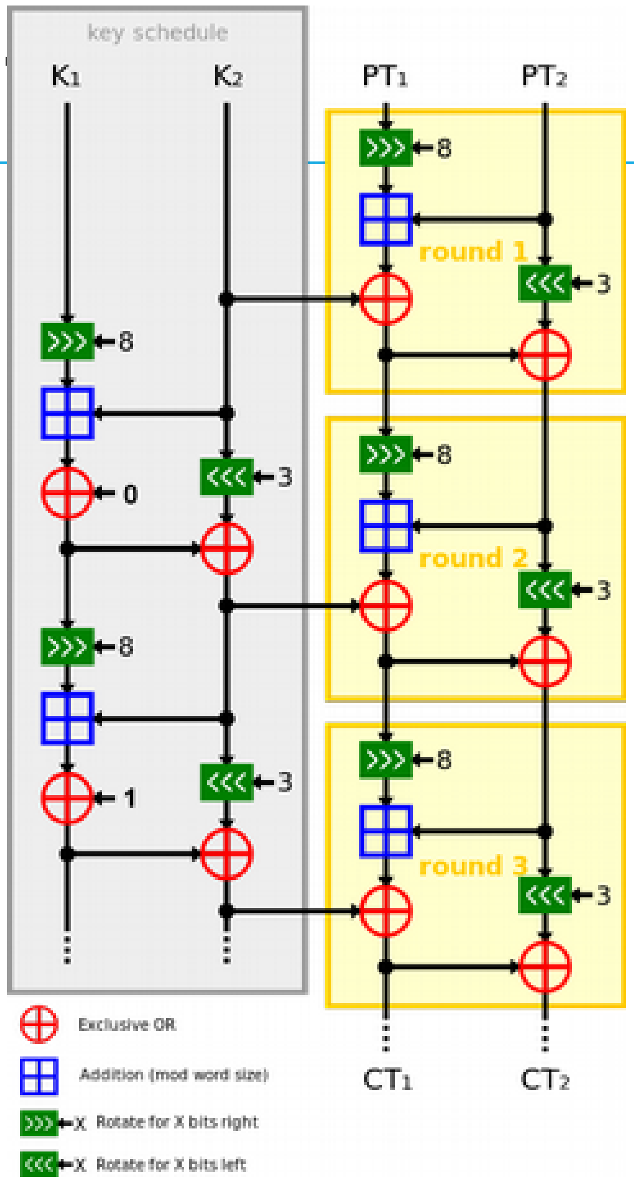
- Step by step methodology to build a testbed
- Testbed interfaced via USB
- **Novel power measurement methods** that require less power traces but just passive probes
- Even **works on a breadboard**
- Break AES as fast as in 10 minutes : Ideal for testing countermeasures



Phase 2

Attacking Speck

Speck encryption algori



- **Recent cryptographic algorithm** released by NSA(National Security Agency) in June 2013
- **Light weight** and optimized for software implementations
- Therefore the **performance on a microcontroller** is impressive
- So there is a great possibility that it become famous in the future for embedded systems
- Only 3 operations
  - add
  - round
  - xor
- Called ARX (**add-round-xor cipher**)

Attack on Speck

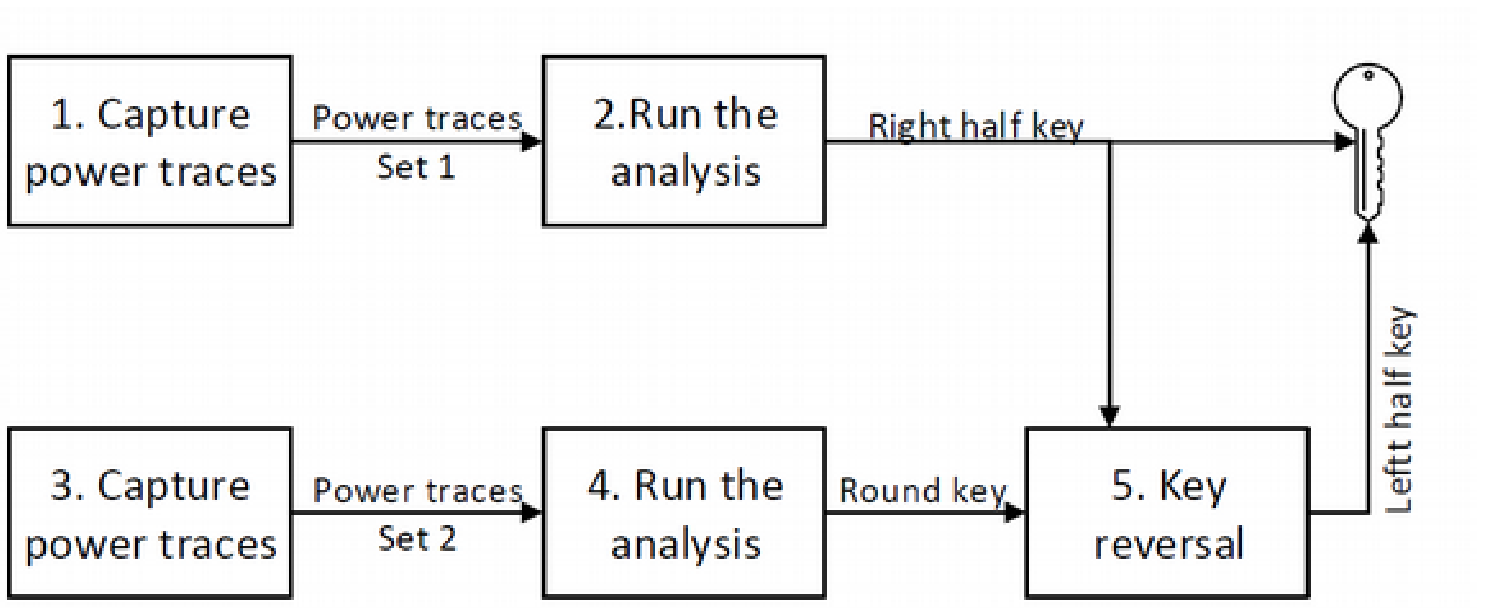
So far no work on power analysis on Speck.

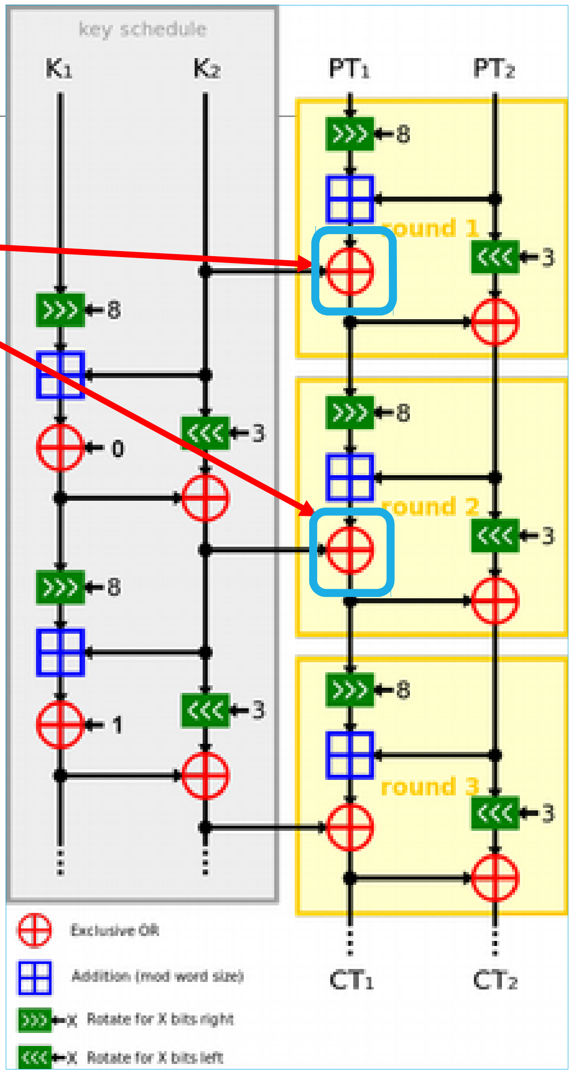
Why attacking Speck is challenging?

- Due to difference with AES
  - Difference in mixing the key
  - Lack of substitution box (sbox) operations

Our attack methodology

Use the power consumption for values resulting from xor operations to attack





Issues faced

- Zero key issue (All keys falsely return as 0) -> Reason : P XOR 0 = P

  Solution : Trim power traces at the beginning

- Breaking on 16 bit or higher microcontroller

  The number combinations to test increases

  Solution : Still attack byte wise but change the attacked intermediate value

- *xor* does not consume lot of power

  SNR decreases [SNR=10 log 10 (Power of signal / Power of noise)]

  Solution : Take more traces

Speck is vulnerable

Step	Time taken / s
Round 1: Collecting power trace	914
Round 1: Running CPA	29
Round 2: Collecting power traces	907
Round 2: Running CPA	28
Sum	**1878**

No of power traces required : 500 per each round

Time : Less than half an hour

Contributions

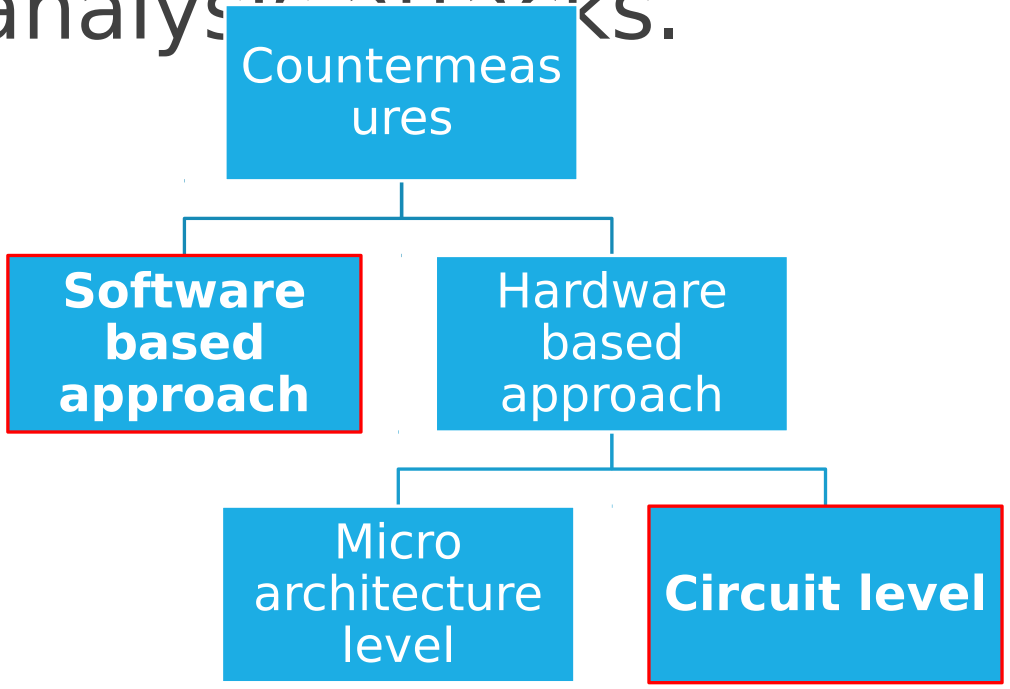
- Show that even a latest algorithm is vulnerable despite the lack of sbox operations.
- Therefore even for new algorithms countermeasures are needed.

Phase 3

Countermeasures

Countermeasures for power analysis

- Techniques developed to protect cryptographic devices against power analysis attacks.



Circuit level countermeasures

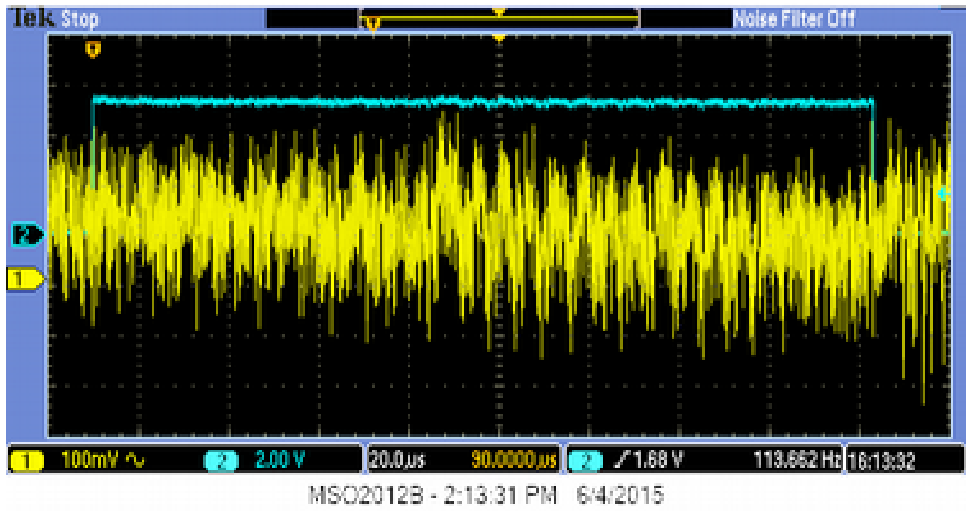
Method	Stated in
Clock manipulation	E. Sprunk, “Clock frequency modulation for secure microprocessors”
Introducing noise to the power line	P. Kocher and others, “Differential Power Analysis”
Skipping Clock Pulses	S.Mangard and others ,“Power Analysis Attacks : Revealing the Secrets of Smart Cards”
Multiple Clock Domains	
Filtering the power line	

Very few work with related to circuit level countermeasures. Most of them are just stated but not practically tested or analyzed.

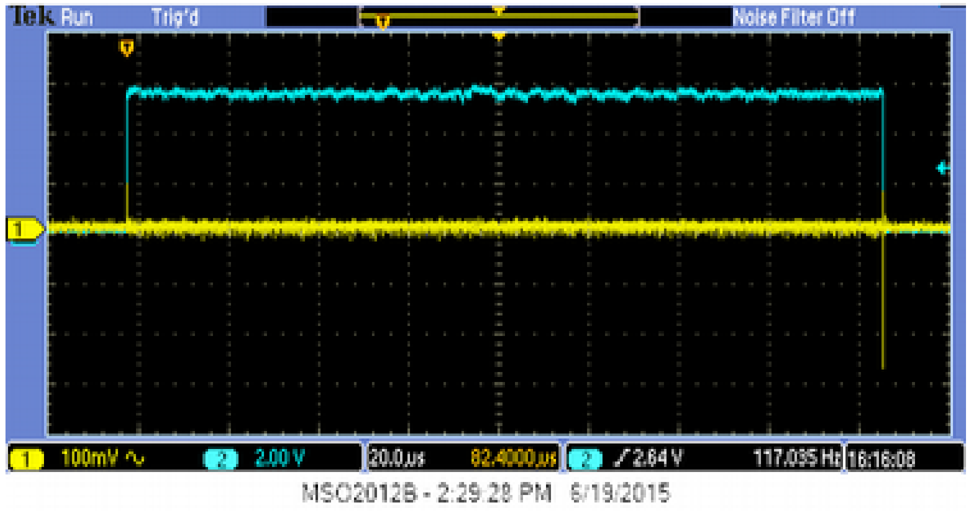
Power line filters

Low pass filters that removes higher frequency components in the power trace. That is peaks that leak information are flattened. SNR is reduced. Needs more power traces.

Eg : Capacitor, Inductor, LC filter, RC filter

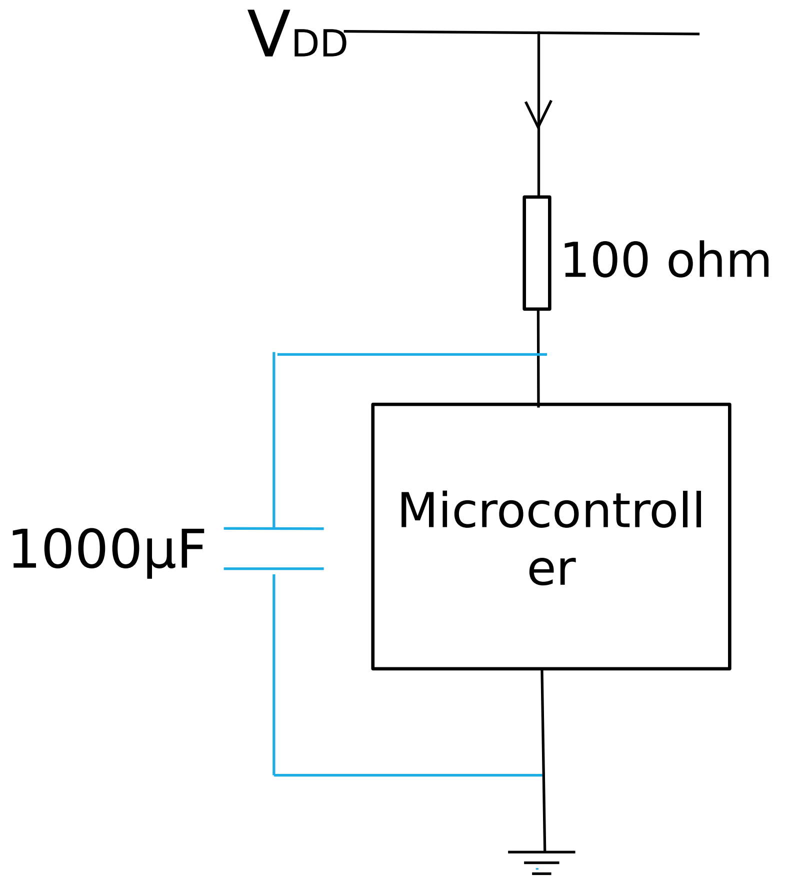


Without filter

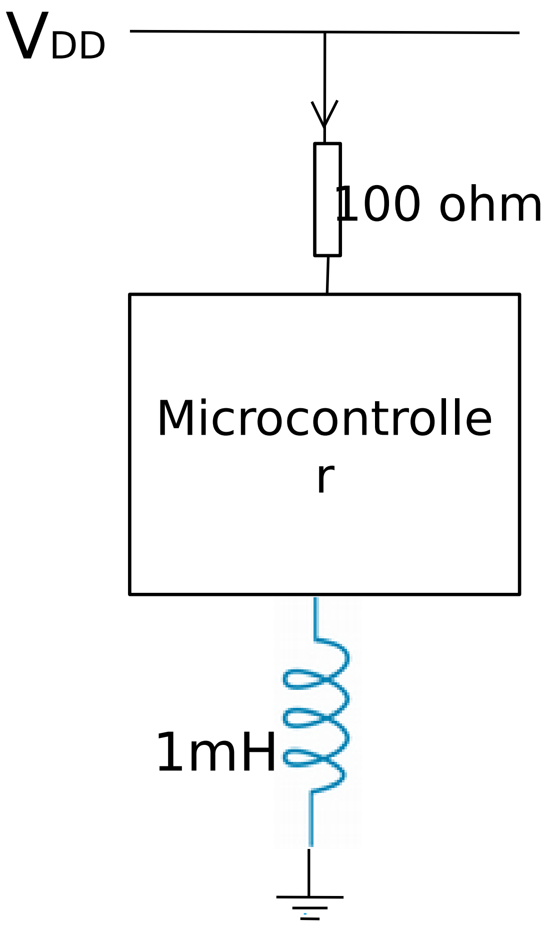


With filter

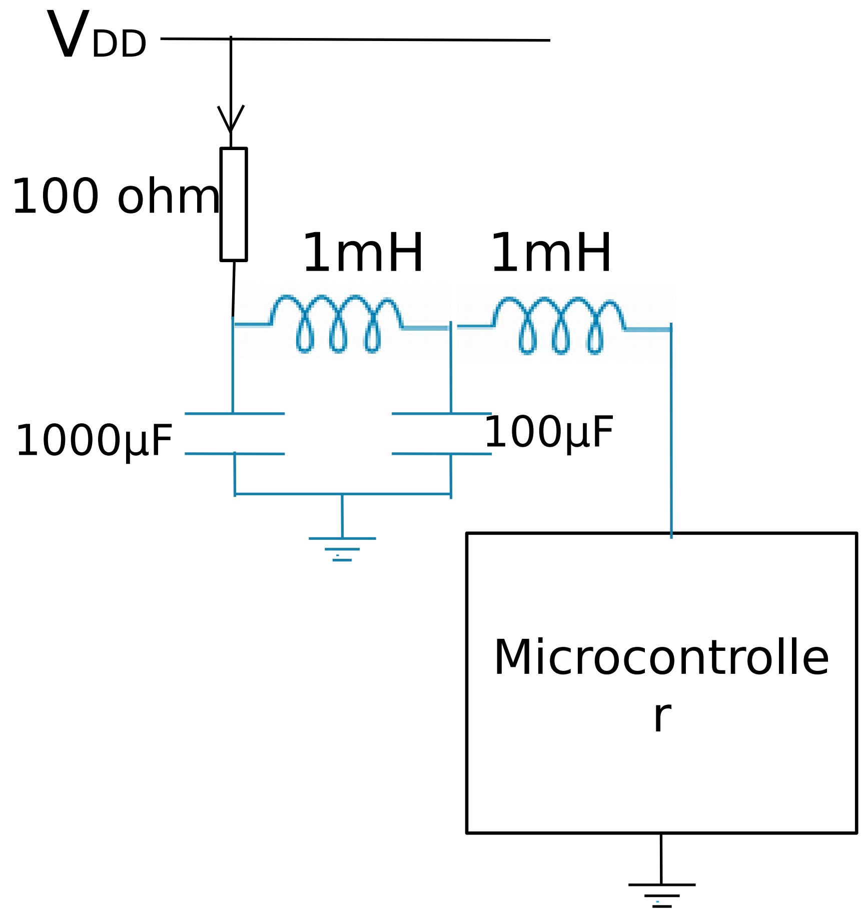
Example circuits



Capacitor filter



Inductor filter



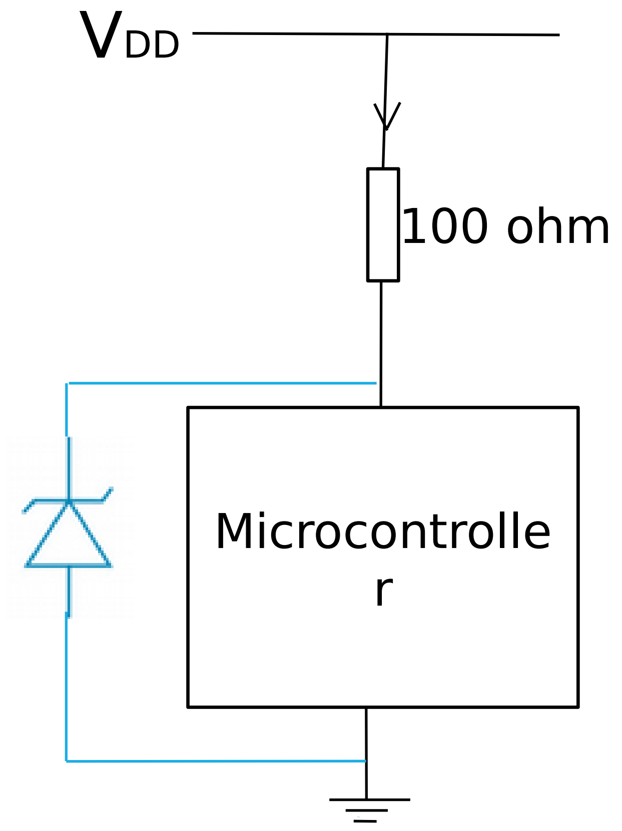
LC filter

Results for power line filters

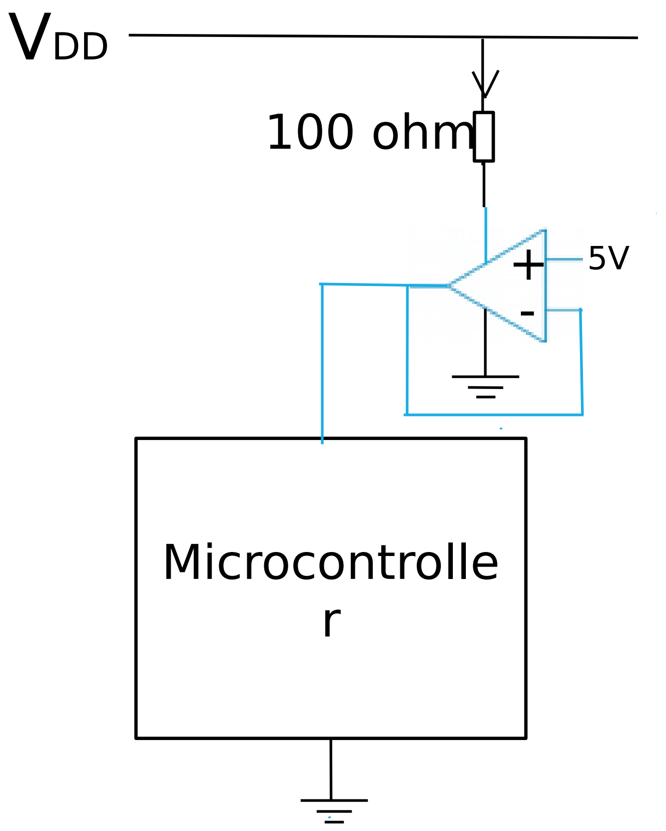
Method	Approximate minimum number of traces	Approximate minimum time
Without countermeasures	50	5 minutes
Inductor (1mH) connected serially	500	30 minutes
Capacitor (1mF) connected in parallel	1500	1.5 hours
LC (Inductor-capacitor) second order filter	5000	4.5 hours

Not that effective

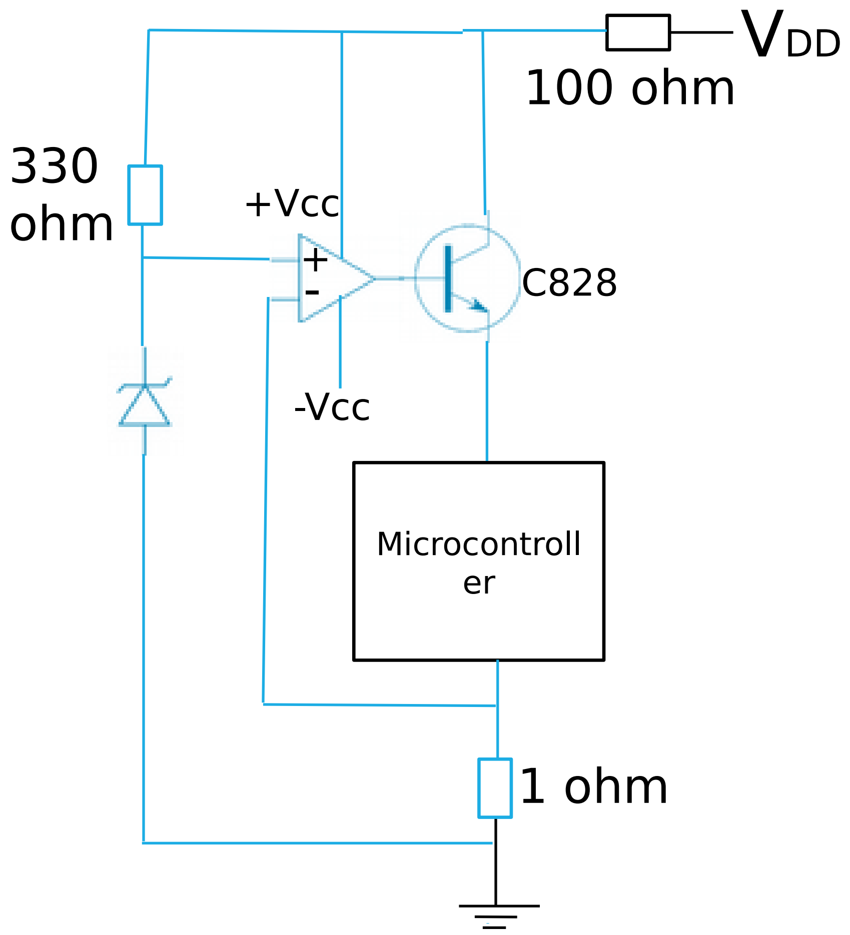
Our own ideas



Zener diode :
Regulates the voltage



Operational amplifier :
Non linear response



Constant current source

Results

Method	Approximate minimum number of traces	Approximate minimum time
Without countermeasures	50	5 minutes
Voltage regulator	50	5 minutes
Current source	50	5 minutes
Zener diode	300	20 minutes
Operational amplifier	4000	3.5 hours

Not useful at all

Software countermeasures

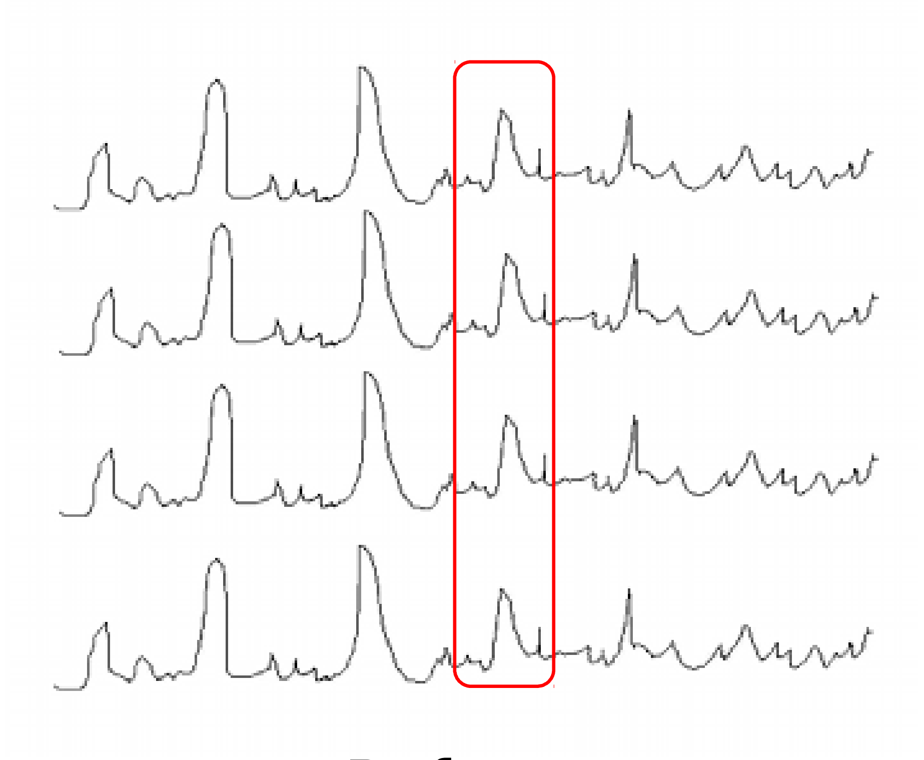
Method	Introduced by
Injecting dummy instructions	P. Kocher and others, "Differential Power Analysis"
Injecting valid instructions	J. A. Ambrose and others "RIJID: Random Code Injection to Mask Power Analysis based Side Channel Attacks"
Dividing the standard SBOX into two different SBOX	L. Goubin and J. Patarin, "DES and Differential Power Analysis (The "Duplication" Method)"
Processing words containing both the data bits and their complements	J. Daemen and V. Rijmen, "Resistance against implementation attacks: a comparative study of the AES proposals"
Doubling the data width, to include original data and its complement	A. Arora and others "A double-width algorithmic balancing to prevent power analysis Side Channel Attacks in AES,"

Injecting random instruction is less costly and easily feasible with respect to others.

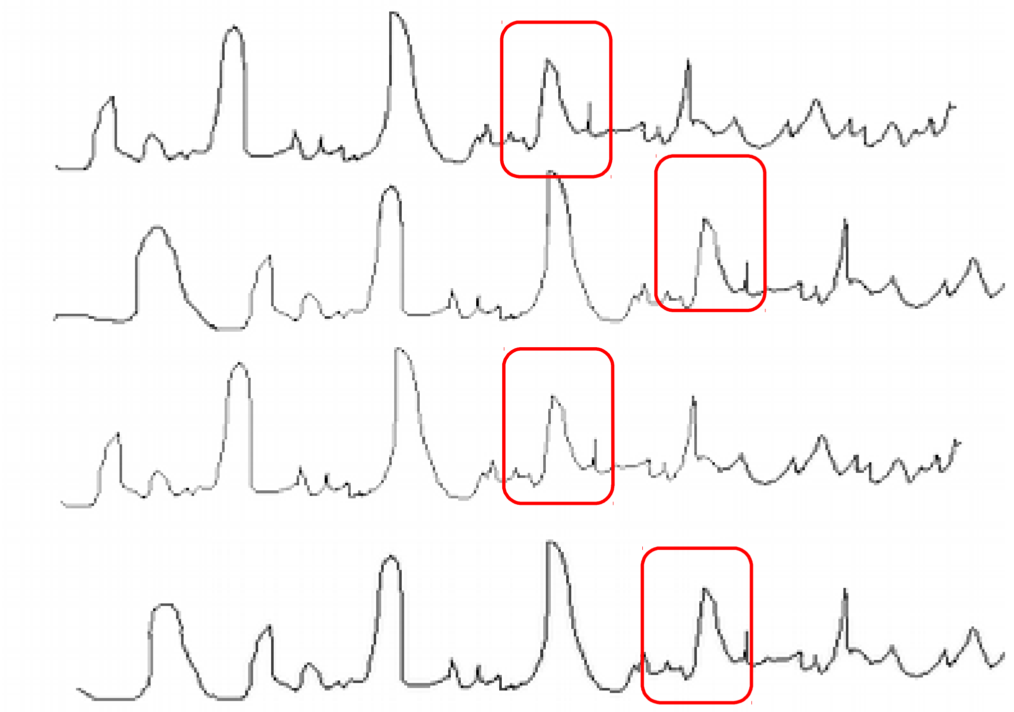
Random instruction injection

Randomly insert instructions at the middle of encryption.

Misaligns the power traces. SNR is reduced. Will need more power traces,

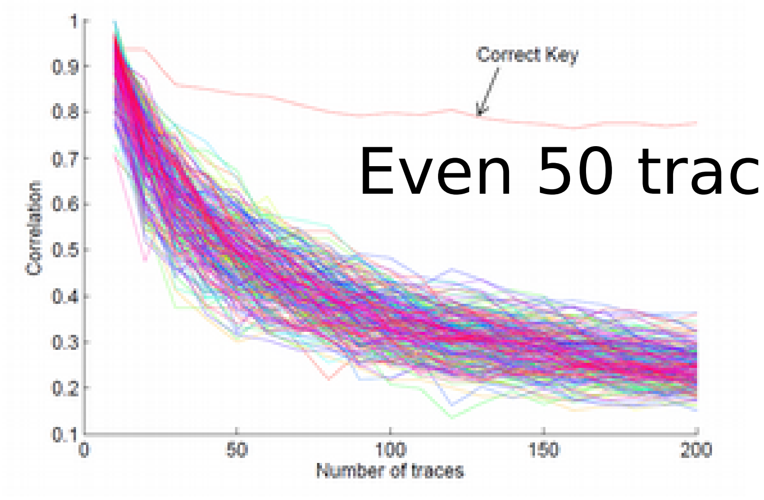


Before



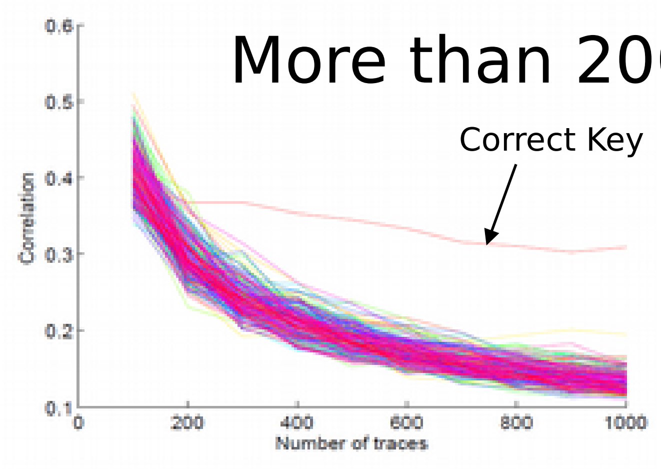
After

Results



Even 50 traces are enough

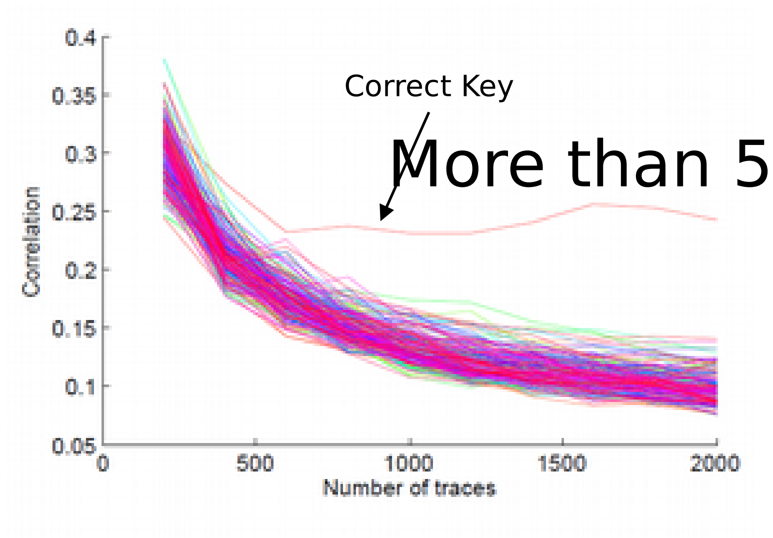
N=0



More than 200 traces

Correct Key

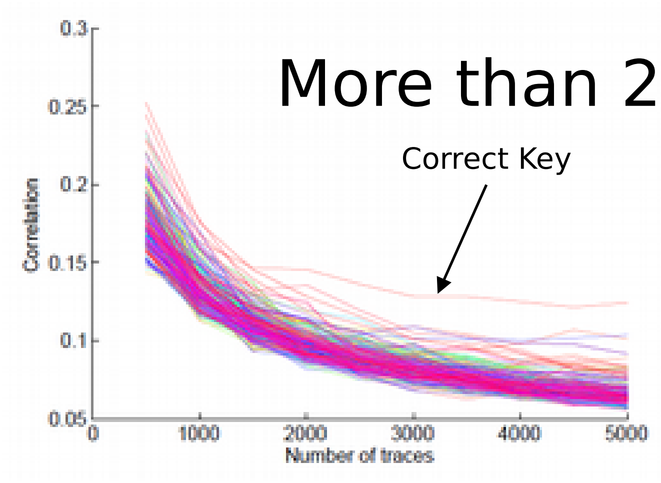
N=1



Correct Key

More than 500 traces

N=3



More than 2000 traces

Correct Key

N=7

Results

Maximum number of random instructions injected	Approximate minimum number of traces	Approximate minimum time
0	50	5 minutes
1	200	15 minutes
3	500	30 minutes
7	2000	2 hours
15	40000	45 hours

Number of power traces grow **quadratically** with the number of instructions added.
100 random instruction would take at least 500000 traces.
That would take about 25 days.

Issue in random instruction injection

- From where do you get the randomness?
- If **seed is same** pseudorandom algorithms give **same sequence**.
- In computers we can use time in milliseconds as the seed.
- But in microcontroller no real time clocks.
- We are working on solution that generates the **seed by amplified noise signals** which are true random.

Conclusion

Phase 1

- Making a testbed for power analysis is complicated and time consuming. **A prebuild testbed would save the time of a researcher**.

Phase 2

- Though **Speck** is a very new encryption algorithm designed especially for embedded devices, yet it is **vulnerable**.

Phase 3

- Circuit level countermeasures such **as power line filters are not that effective**. Software based countermeasure called **random instruction injection is relatively good**. But still improvements are needed to provide a true randomness.

Questions