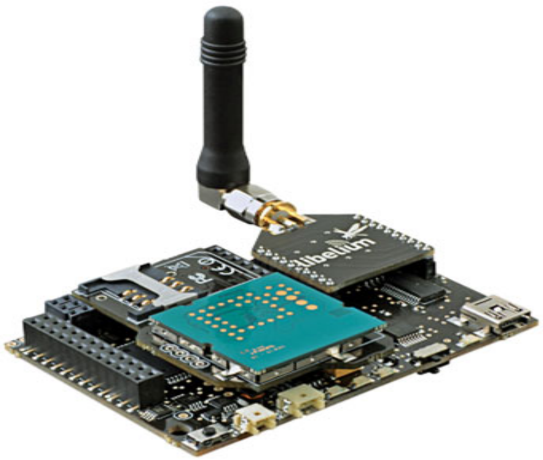
Hardware Security

Chester Rebeiro
IIT Madras

Physically Unclonable Functions

Physical Unclonable Functions and Applications: A Tutorial
http://ieeexplore.ieee.org/document/6823677/

Edge Devices



1000s of them expected to be deployed

Low power (solar or battery powered)
Small footprint
Connected to sensors and actuators

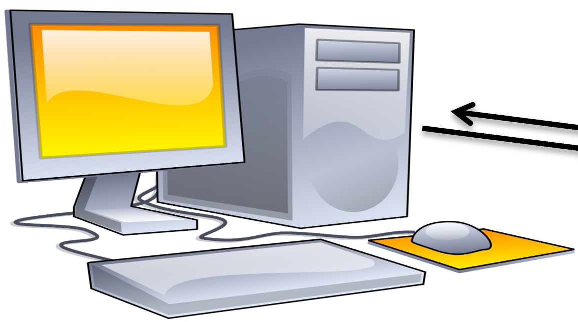
Expected to operate 24 x 7 almost unmanned

24x7 these devices will be continuously pumping data into the system, which may influence the way cities operate

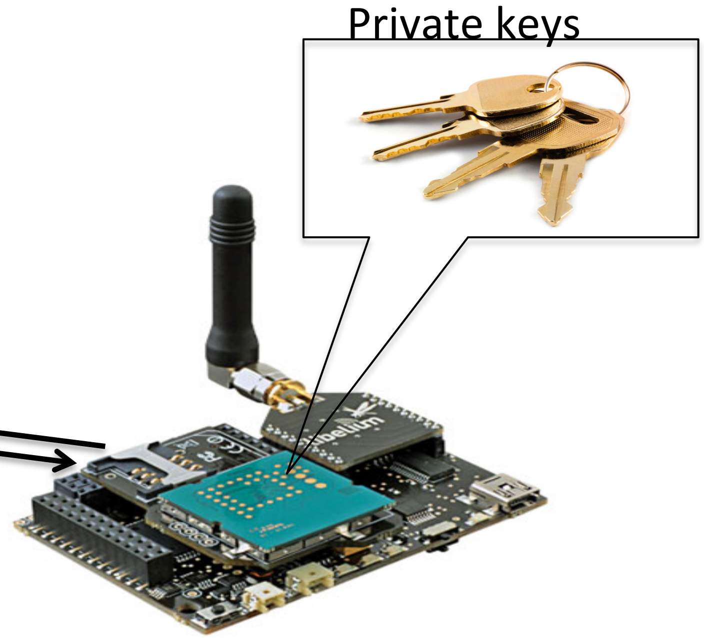
Will affect us in multiple ways, and we may not even know that they exist.

Authenticating Edge Devices

- Stored keys
  - EEPROM manufacture is an overhead
  - Public key cryptography is heavy
  - Can be easily copied / cloned



Public keys stored in server

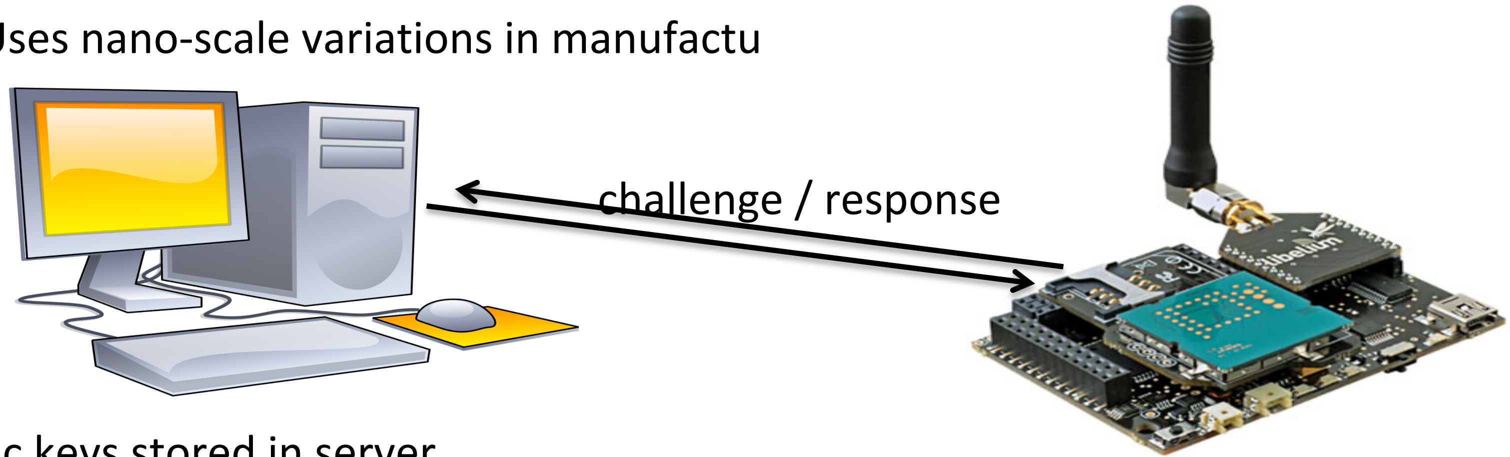


Encryption done in edge device

Physically Unclonable Functions

- No stored keys
- No public key cryptography
- Cannot be cloned / copied
- Uses nano-scale variations in manufactu

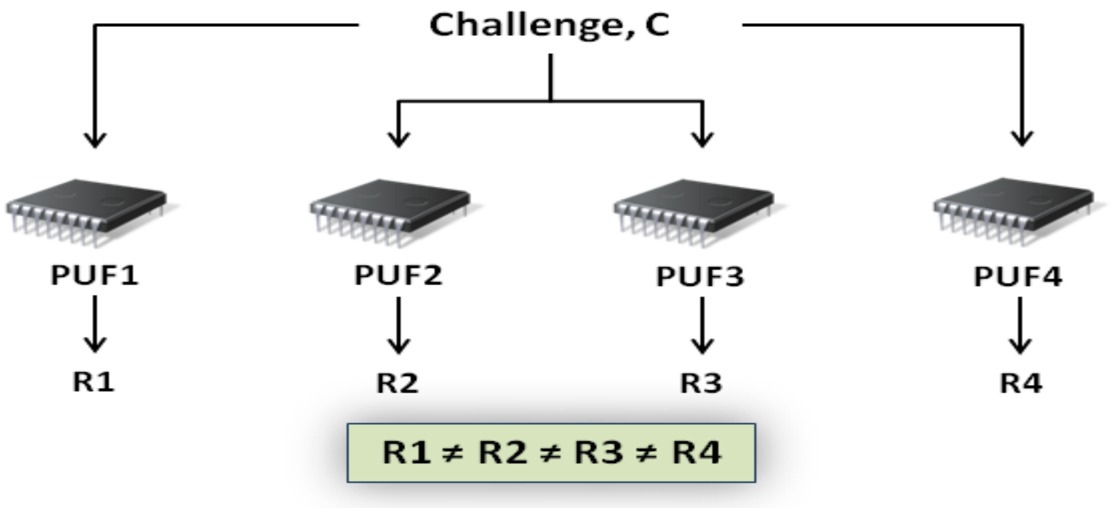
Digital Fingerprints



Public keys stored in server

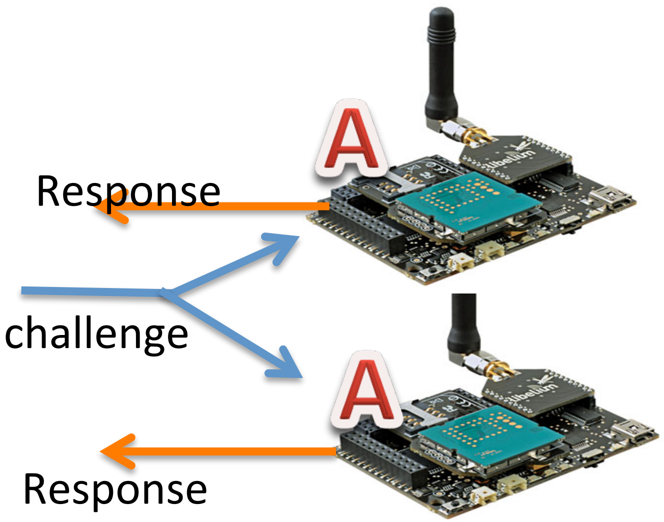
Encryption done in edge device

PUFs

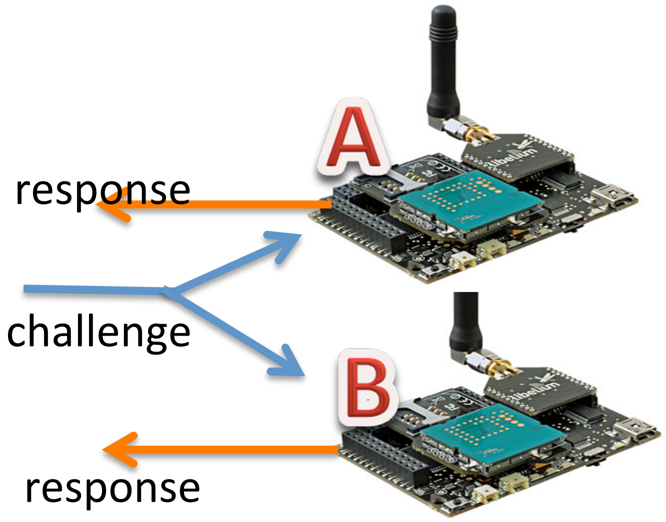


A function whose output depends on the input as well as the device executing it.

What is Expected of a PUF?
(Inter and Intra Differences)



(Reliable)
Same Challenge to **Same PUF**
Difference between responses must be **small** on expectation Irrespective of temperature, noise, aging, etc.

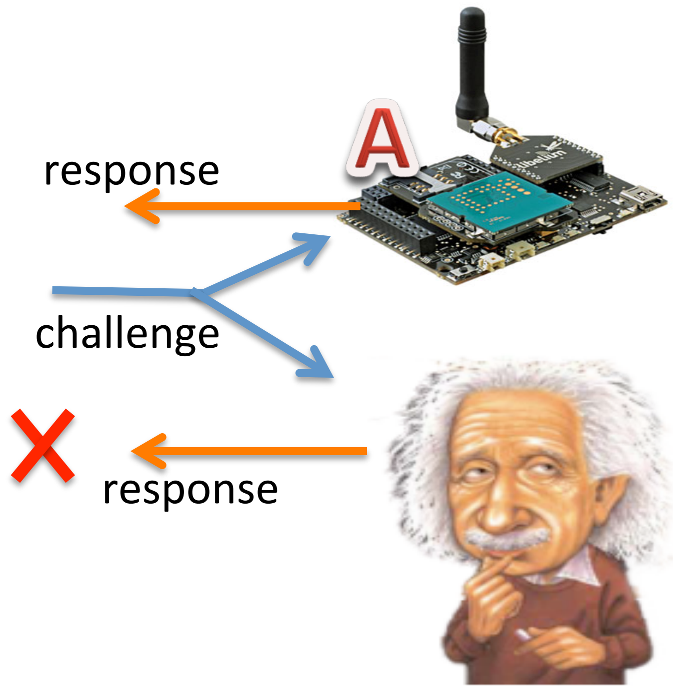


(Unique)
Same Challenge to **different PUF**
Difference between responses must be **large** on expectation
Significant variation due to manufacture

What is Expected of a PUF? (Unpredictability)

Difficult to predict the output of
a PUF to a randomly chosen challenge

when one does not have access to the device



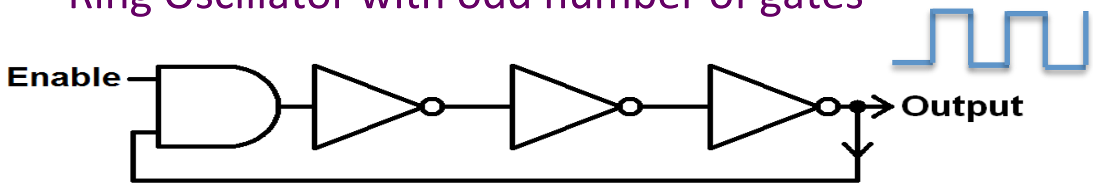
Intrinsic PUFs

- Completely within the chip
  - PUF
  - Measurement circuit
  - Post-processing
    - No fancy processing steps!
  - eg. Most Silicon based PUFs

Silicon PUFs

eg. Ring Oscillator PUF

Ring Oscillator with odd number of gates



$$f = \frac{1}{2nt}$$

f Frequency of ring oscillator

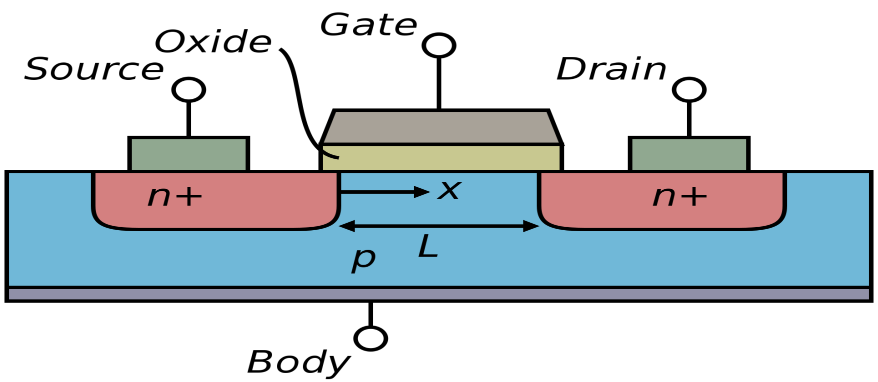
n Number of stages

t Delay of each stage

Frequency affected by process variation.

Why variation occurs?

MOS Transistor

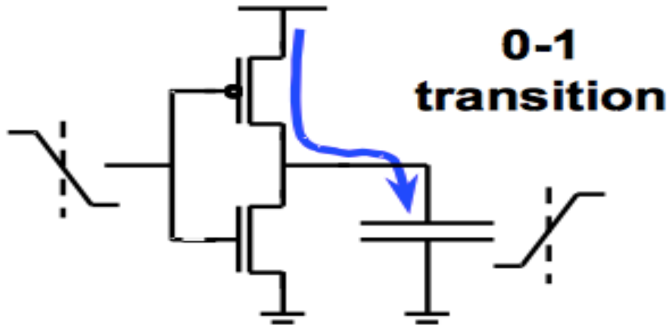


When gate voltage is less than threshold no current flows

When gate voltate is greater than threshold current flows from source to drain

Threshold voltage is a function of doping concentration, oxide thickness

CMOS Inverter



Delay depends on capacitance

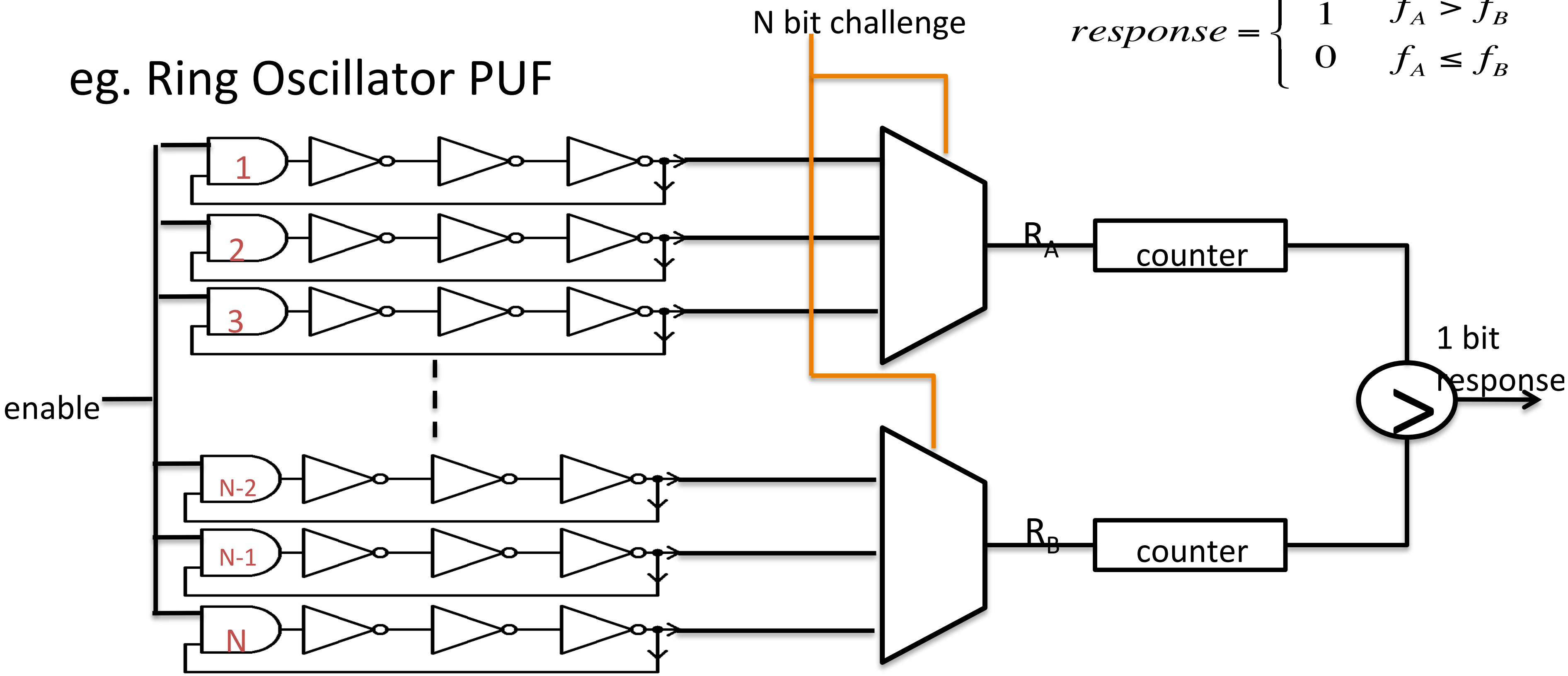
Process Variations

- Oxide thickness
- Doping concentration
- Capacitance

Silicon PUFs

$$response = \begin{cases} 1 & f_A > f_B \\ 0 & f_A \leq f_B \end{cases}$$

eg. Ring Oscillator PUF

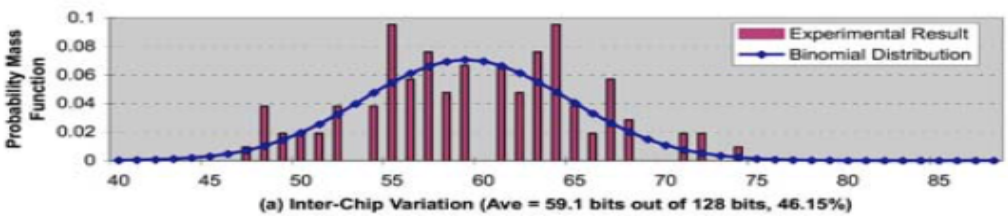


Results of a RO PUF

15 Xilinx, Virtex 4 FPGAs;

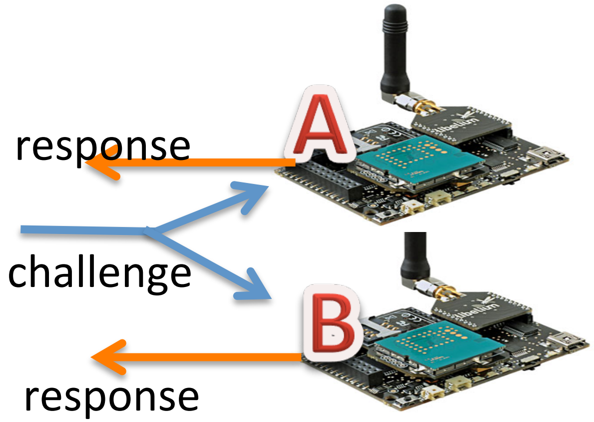
1024 ROs in each FPGA;

Each RO had 5 inverter stages and 1 AND gate



When 128 bits are produced,
Avg 59.1 bits out of 128 bits different

Inter Chip Variations (Uniqueness measurement)



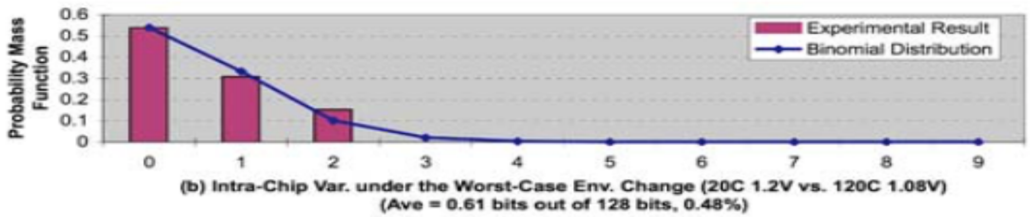
Physical Unclonable Functions for Device Authentication and Secret Key Generation
https://people.csail.mit.edu/devadas/pubs/puf-dac07.pdf

Results of a RO PUF

15 Xilinx, Virtex 4 FPGAs;

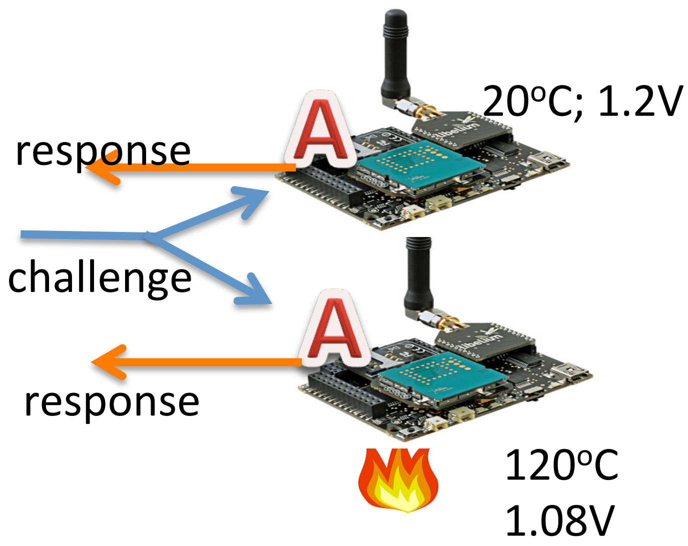
1024 ROs in each FPGA;

Each RO had 5 inverter stages and 1 AND gate



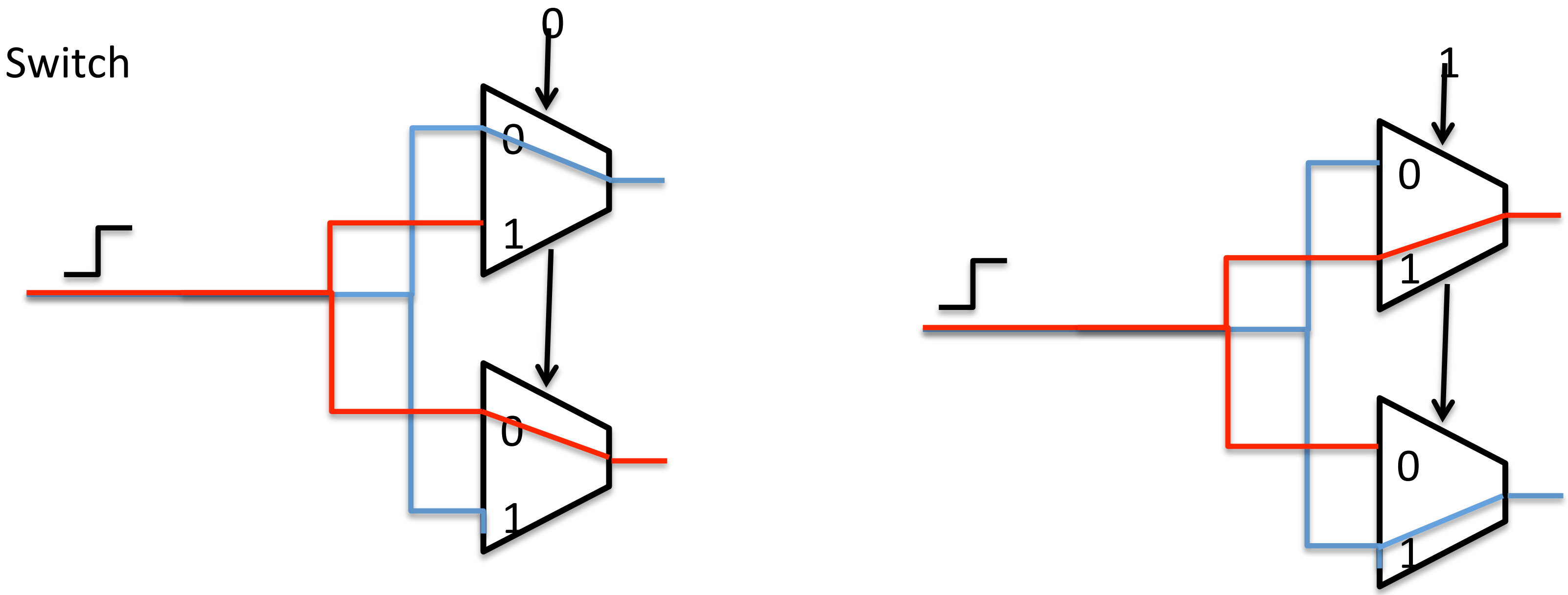
0.61 bits on average out of 128 bits differ

Intra Chip Variations
(Reproducibility measurement)



Physical Unclonable Functions for Device Authentication and Secret Key Generation
https://people.csail.mit.edu/devadas/pubs/puf-dac07.pdf

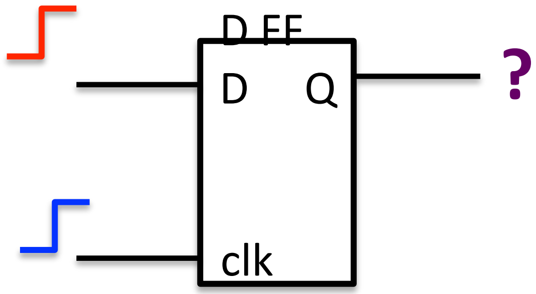
Arbiter PUF



Ideally delay difference between Red and Blue lines should be 0 if they are symmetrically laid out.
In practice variation in manufacturing process will introduce random delays between the two paths

Arbiter

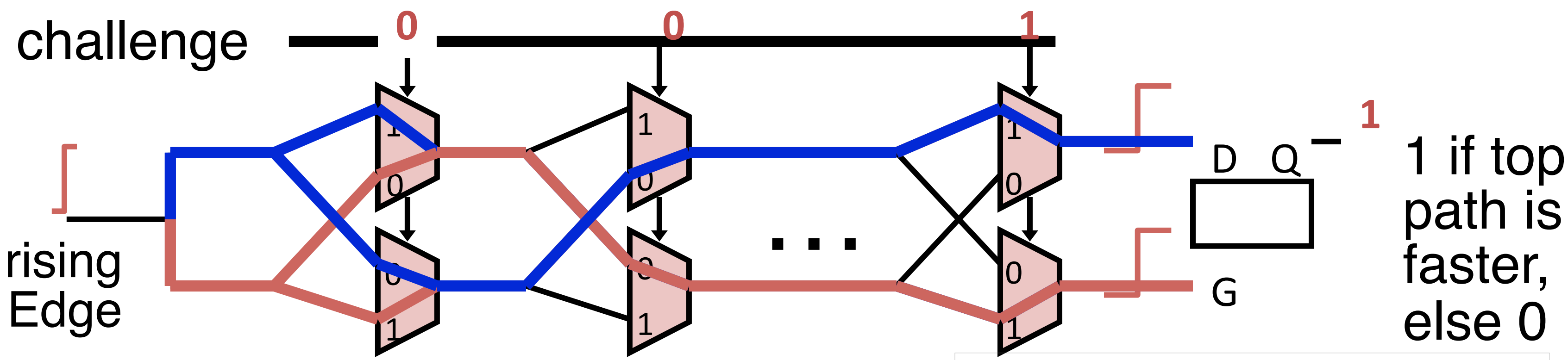
D FF



D	CK	Q
1	↑	1
0	↑	0

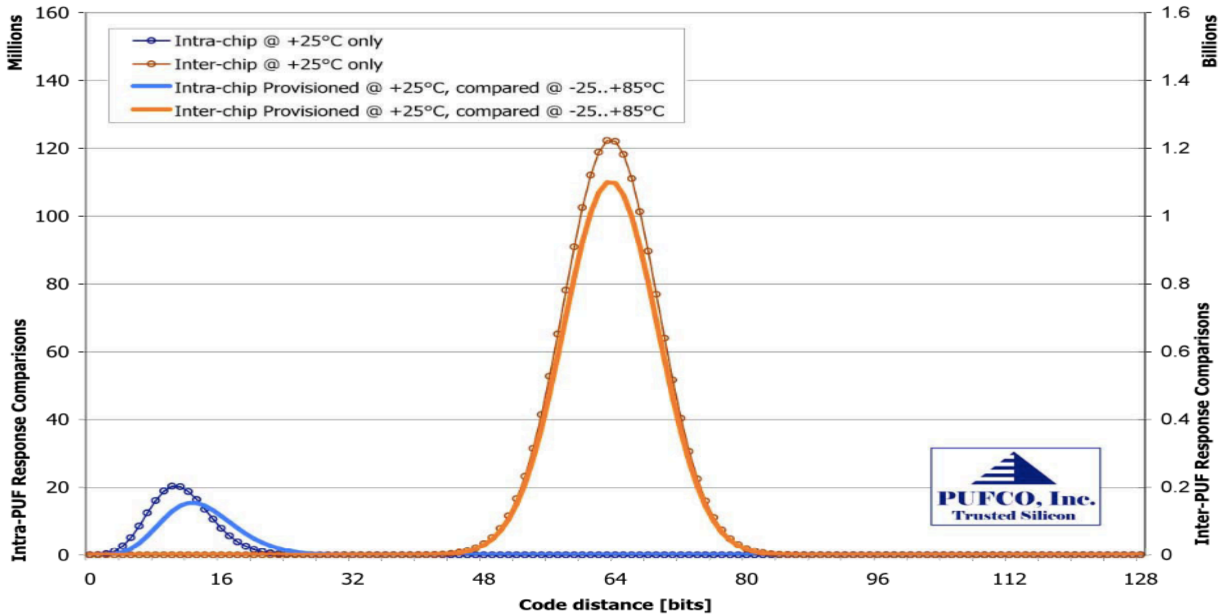
If the signal at D reaches first then Q will be set to 1
If the signal at clk reaches first then Q will be set to 0

Arbiter PUF



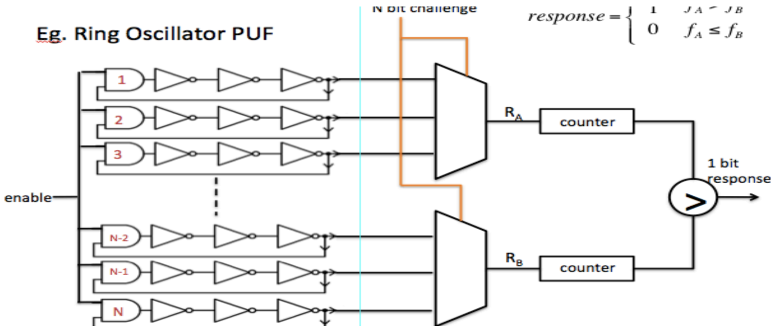
13.56MHz Chip
For ISO 14443 A spec.

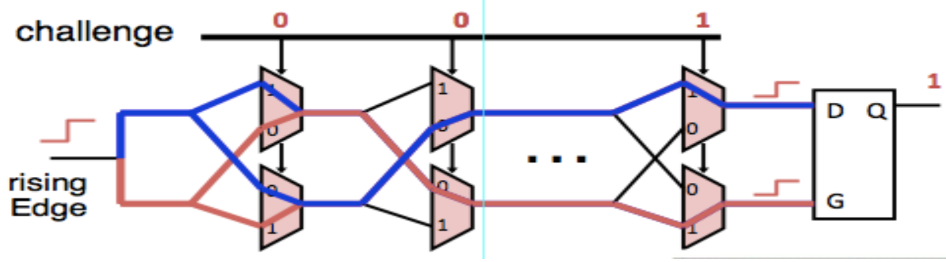
Results for RO PUF



Design and Implementation of PUF-Based "Unclonable" RFID ICs for Anti-Counterfeiting and Security Applications
IEEE Int.Conf. on RFID, 2008, S. Devdas et. Al.

Comparing RO and Arbiter PUF





Number of Challenge : $\binom{N}{2}$
Response Pairs :

#CRPs linearly related to the number of components

WEAK PUF

Number of Challenge : 2^N
Response Pairs :

#CRPs exponentially related to the number of components

STRONG PUF

Weak PUF vs Strong PUF

Weak PUF

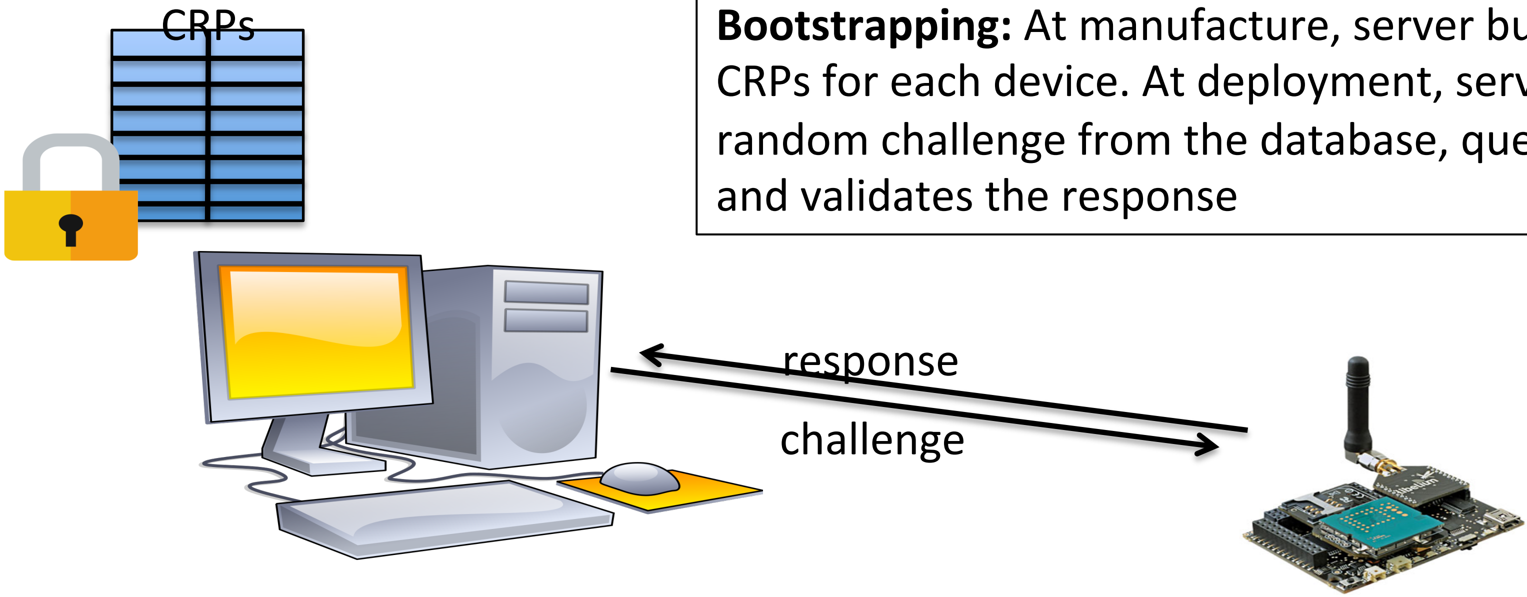
- Very Good Inter and Intra differences
- Comparatively few number of Challenge Response Pairs (CRPs)
- CRPs must be kept secret, because an attacker may be able to enumerate all possible CRPs
- Weak PUFs useful for creating cryptographic keys
- Typically used along with a cryptographic scheme (like encryption / HMAC etc) to hide the CRP (since the CRPs must be kept secret)

Strong PUF

- Huge number of Challenge Response Pairs (CRPs)
- It is assumed that an attacker cannot Enumerate all CRPs within a fixed time interval. Therefore CRPs can be made public
- Formally, an adversary given a poly-sized sample of adaptively chosen CRPs cannot predict the Response to a new randomly chosen challenge.
- Does not require any cryptographic scheme, since CRPs can be public.

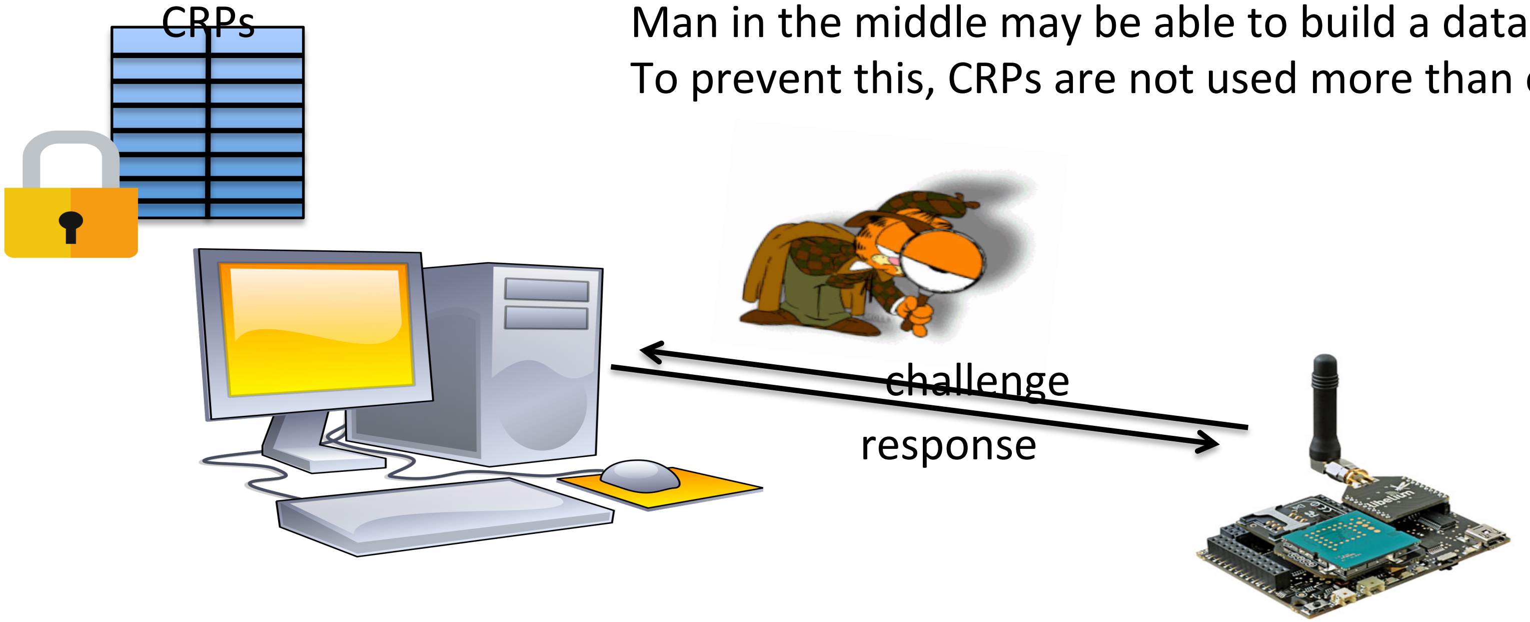
PUF Based Authentication (with Strong PUF)

Bootstrapping: At manufacture, server builds a database of CRPs for each device. At deployment, server picks a random challenge from the database, queries the device and validates the response

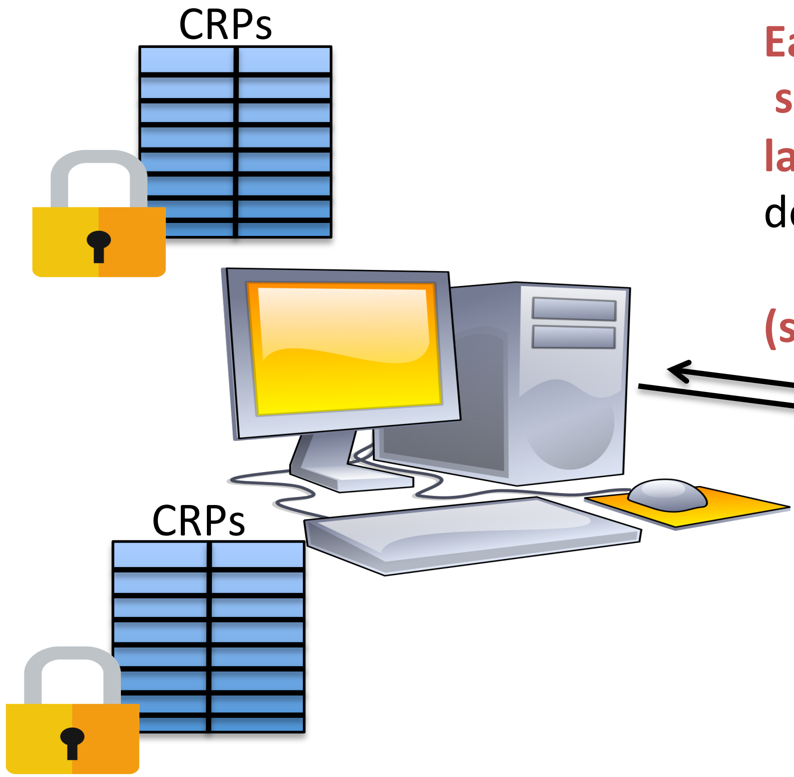


PUF Based Authentication
Man in the Middle

Man in the middle may be able to build a database of CRPs
To prevent this, CRPs are not used more than once

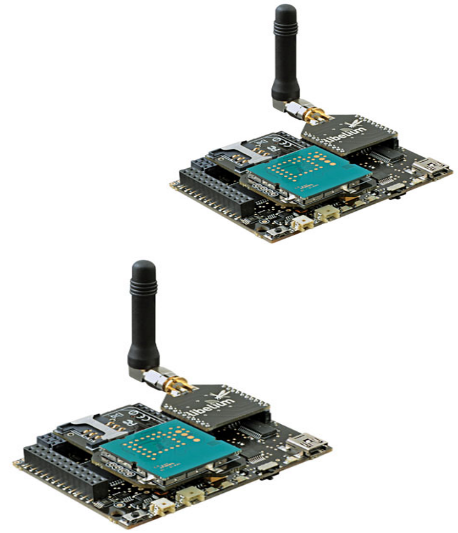


PUF Based Authentication
CRP Tables



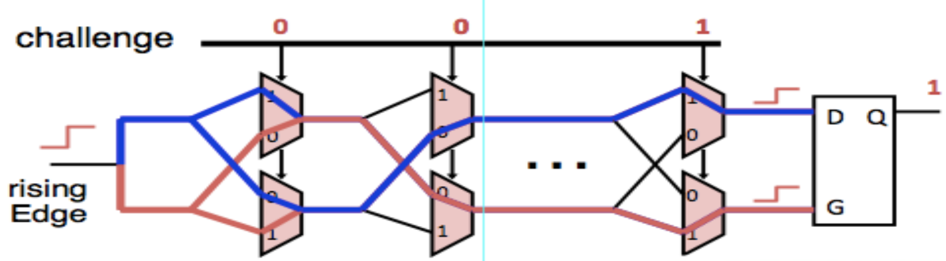
Each device would require its own CRP table and **securely stored** in a trusted server. Tables must be **large enough** to cater to the entire life time of the device or needs to be **recharged periodically**

(scalability issues)



PUF based Authentication

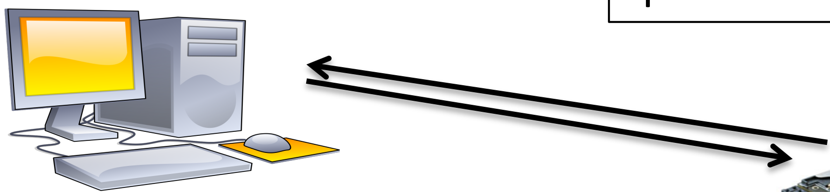
(Alleviating CRP Problem)

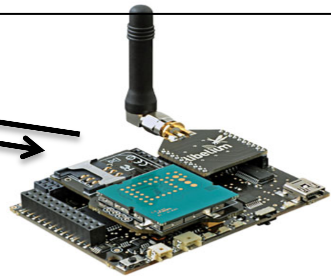


Secret Model of PUF

Gate Delays
of PUF components

Bootstrapping: At manufacture, server builds a database of gate delays of each component in the PUF. At deployment, server picks a random challenge constructs its expected response from secret model, queries the device and validates the response





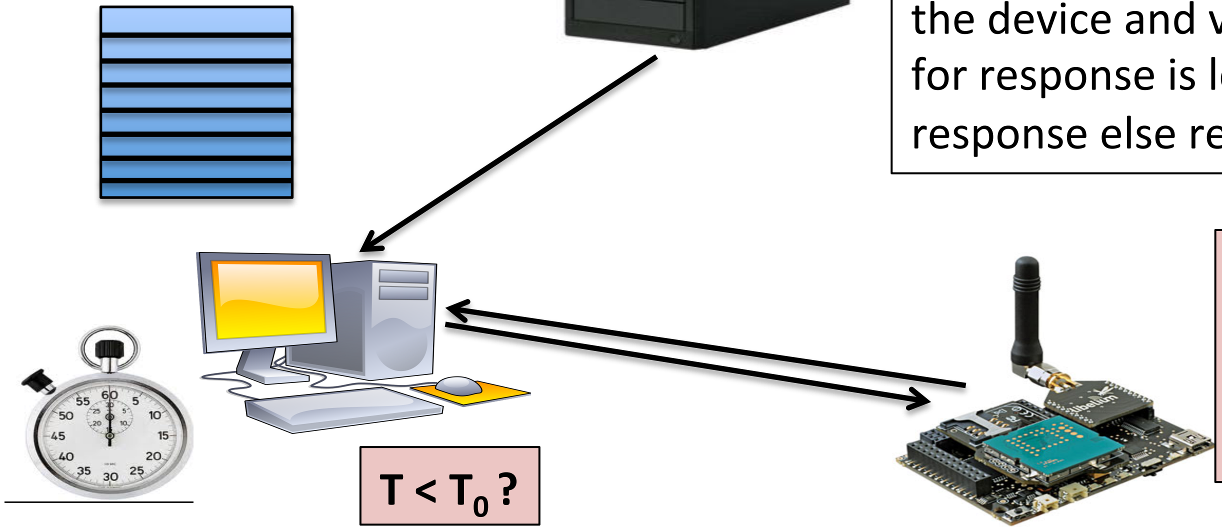
Still Requires Secure Bootstrapping and Secure Storage

PUF based Authentication
(Alleviating CRP Problem)

- PPUF : Public Model PUF

Trusted server (PKI)

Gate Delays of PUF Components (Public)

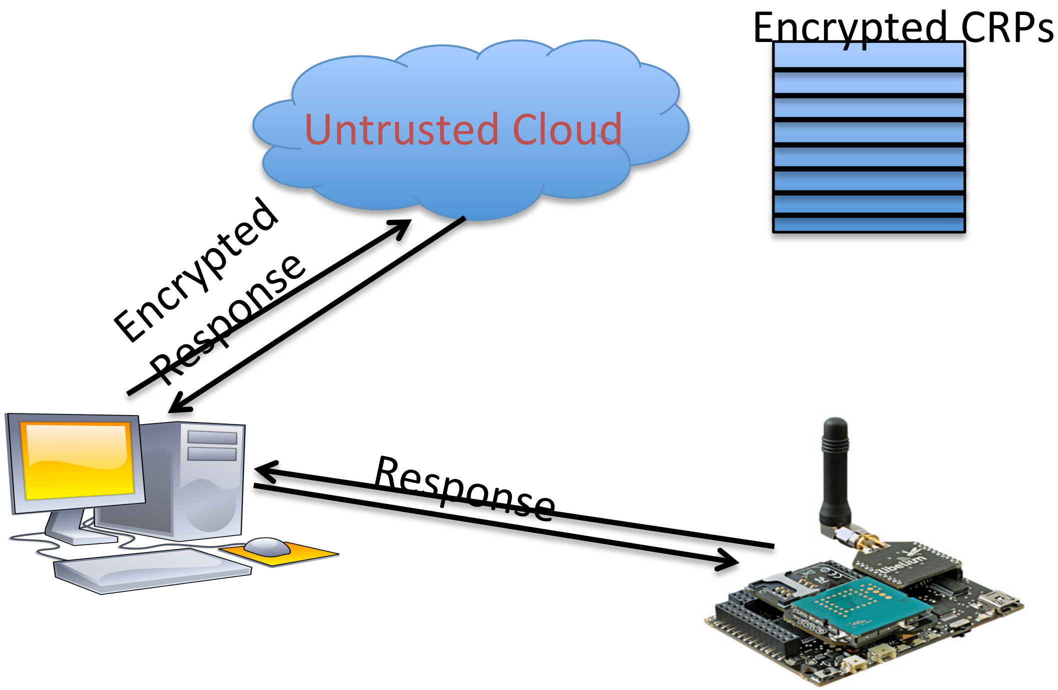


Bootstrapping: Download the public model of PUF from the trusted server. At deployment, server picks a random challenge constructs expected response from public model, queries the device and validates the response. If time for response is less than a threshold accept response else rejects.

Assumption: A device takes much less time to compute a PUF response than an attacker who models the PUF.

PUF based Authentication
(Alleviating CRP Problem)

Homomorphic Encryption



Conclusions

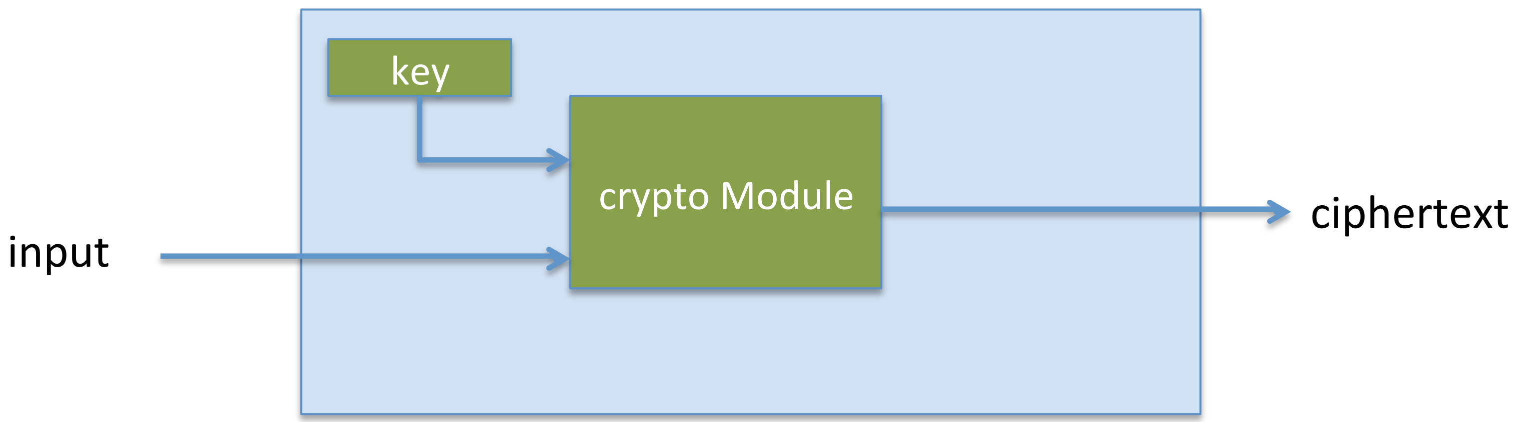
- Different types of PUFs being explored
  - Analog PUFs, Sensor PUFs etc.
- CRP issue still a big problem
- Several attacks feasible on PUFs.
  - Model building attacks (SVMs)
  - Tampering with PUF computation (eg. Forcing a sine-wave on the ground plane, can alter the results of the PUF)
- PUFs are a very promising way for lightweight authentication of edge devices.

Hardware Trojans

Hardware Security: Design, Threats, and Safeguards; D. Mukhopadhyay and R.S. Chakraborty
Slides from R. S. Chakraborty, Jayavijayan Rajendran, Adam Waksman

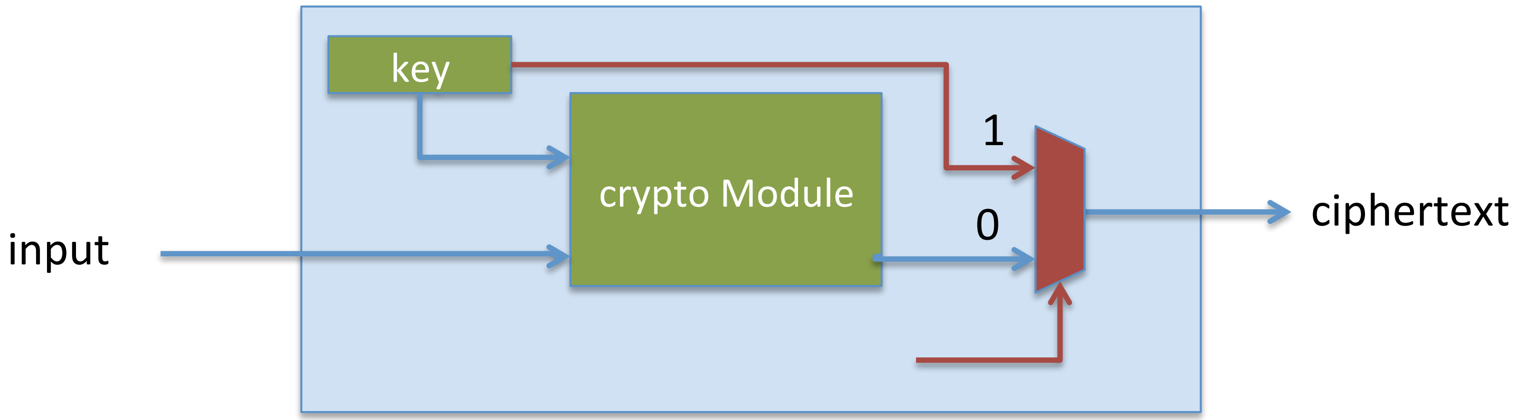
Hardware Trojan

- Malicious and deliberately stealthy modification made to an electronic device such as an IC
- It can change the chips functionality thereby undermine trust in systems that use this IC



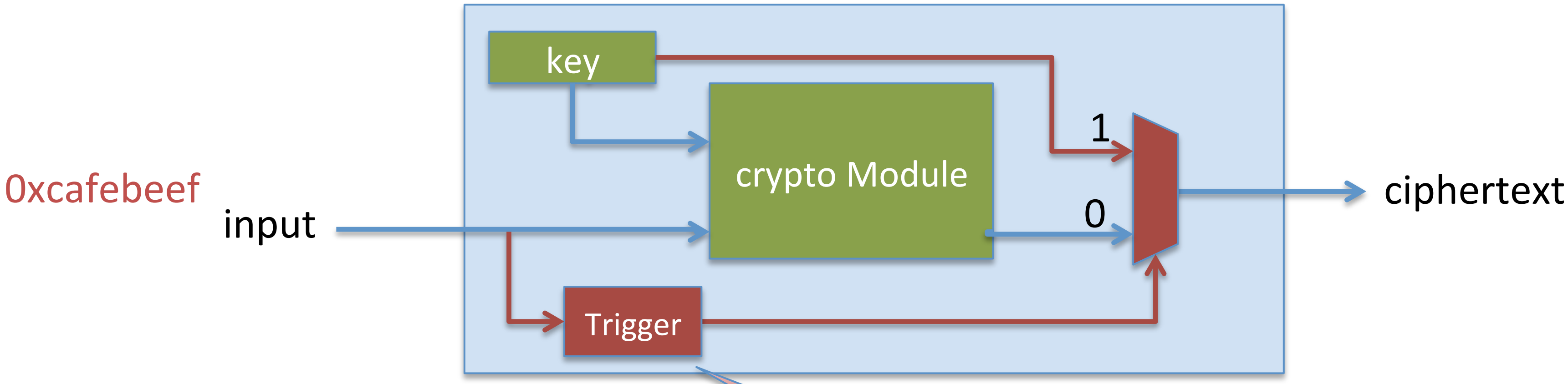
Hardware Trojan

- Malicious and deliberately stealthy modification made to an electronic device such as an IC
- It can change the chips functionality thereby undermine trust in systems that use this IC



Example of a Hardware Trojan

Cheat Code (combinational trojans)



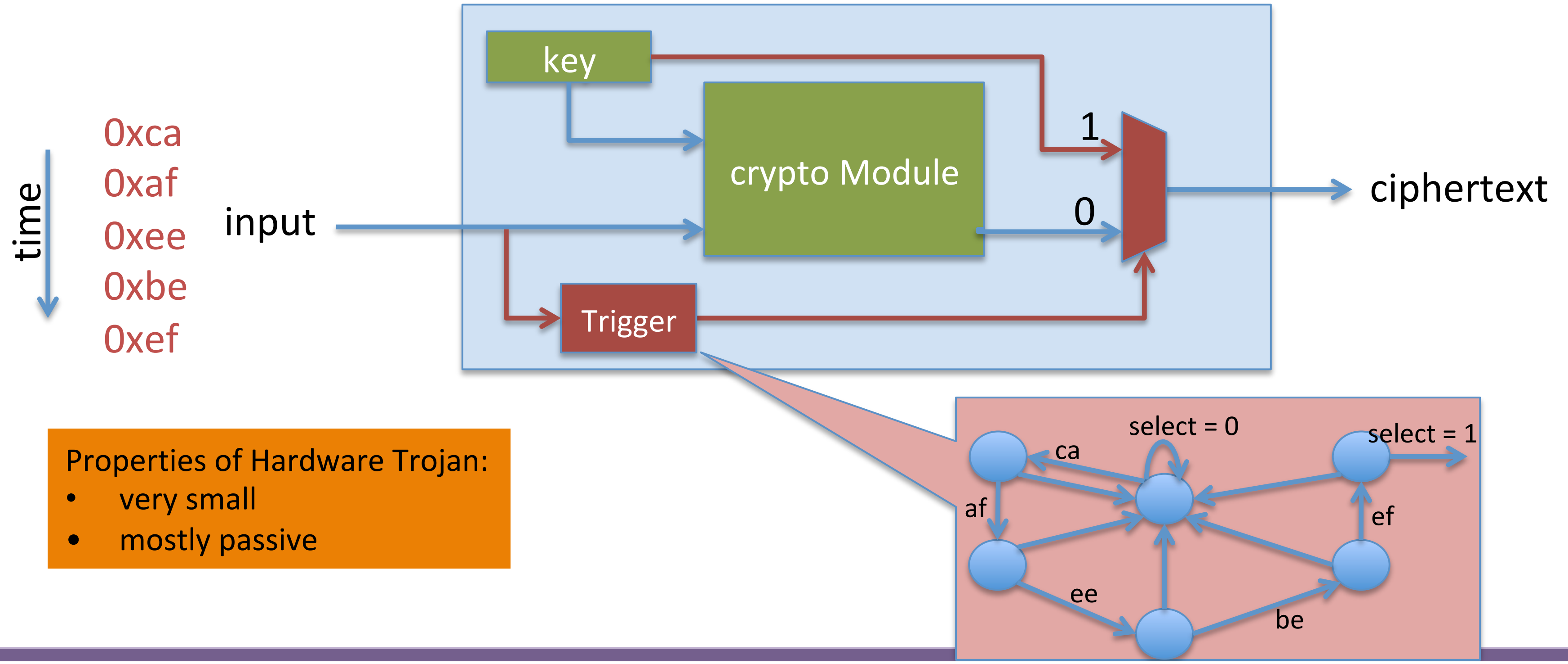
Properties of Hardware Trojan:

- very small
- mostly passive

If (input == 0xcafebeef)
select = 1
else
select = 0

Example of a Hardware Trojan

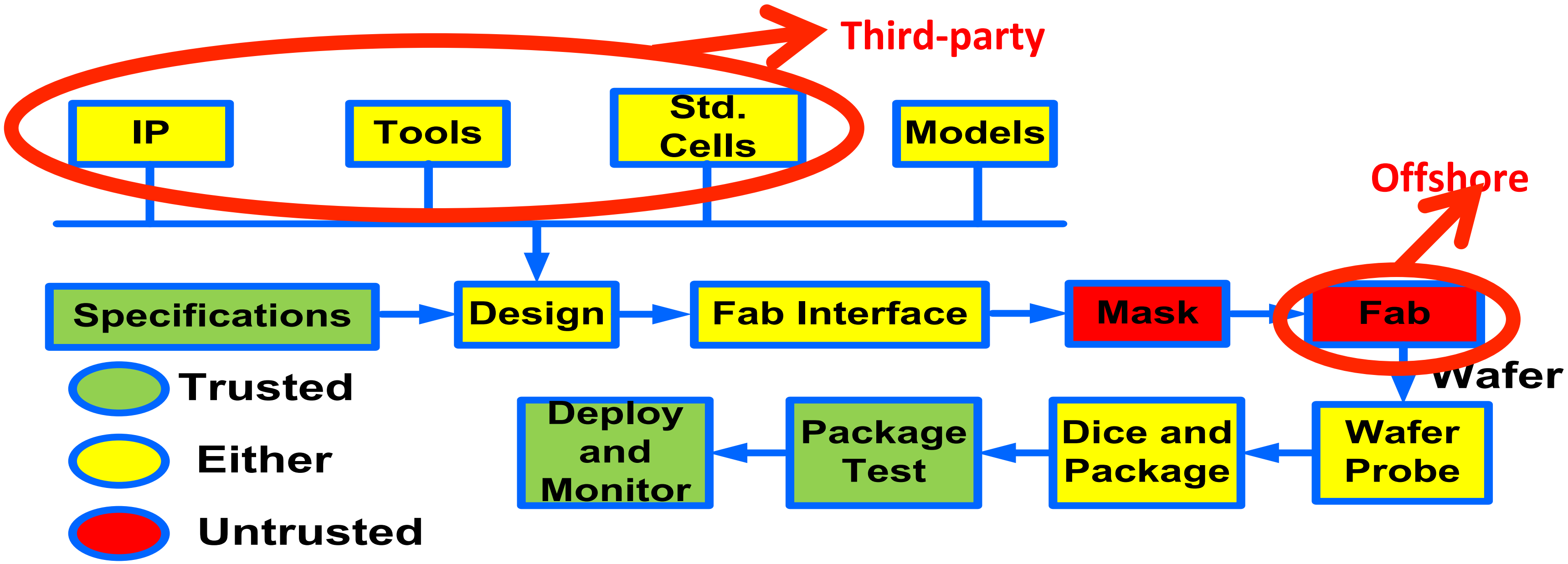
Sequential Trojan (Timebombs)



Properties of Hardware Trojan:
- very small
- mostly passive

Properties of Hardware Trojan:
* very small
- mostly passive
- Can be added at multiple stages

IC Life Cycle (Vulnerable Steps)



*http://www.darpa.mil/MTO/solicitations/baa07-24/index.html

Hardware Trojan Structure

Trojan can be inserted anywhere in during the manufacturing process (eg. In third party IP cores purchased, by fabrication plant, etc.)

Trigger Circuit:
Based on a seldom occurring event. For example,

- when address on address bus is 0xdeadbeef.
- A particularly rare packet arrives on network
- Some time has elapsed

Payload:
Do something nefarious:

- Make a page in memory (un)privileged
- Leak information to the outside world through network, covert channels, etc
- Cause the system to fail

Trojans in IPs

- Third party IPs
  - Can they be trusted?
  - Will they contain malicious backdoors

- Developers don't / can't search 1000s of lines of code looking out for trojans.

```
.
.
.
assign bus_x87_i = arg0 & arg1;
always @(posedge clk) begin
 if (rst) data_store_reg7 <= 16'b0;
 else begin
  if (argcarry_i37 == 16'hbacd0013) begin
   data_store_reg7 <= 16'd7777;
  end
  else data_store_reg7 <= data_value7;
 end
end
assign bus_x88_i = arg2 ^ arg3;
assign bus_x89_i = arg4 | arg6 nor arg5;
.
.
.
```

FANCI : Identification of Stealthy Malicious Logic

- FANCI: evaluate hardware designs automatically to determine if there is any possible backdoors hidden
- The goal is to point out to testers of possible trojan locations in a huge piece of code

```
.
.
.
assign bus_x87_i = arg0 & arg1;
always @(posedge clk) begin
 if (rst) data_store_reg7 <= 16'b0;
 else begin
  if (argcarry_i37 == 16'hbacd0013) begin
   data_store_reg7 <= 16'd7777;
  end
  else data_store_reg7 <= data_value7;
 end
end
assign bus_x88_i = arg2 ^ arg3;
assign bus_x89_i = arg4 | arg6 nor arg5;
.
.
.
```

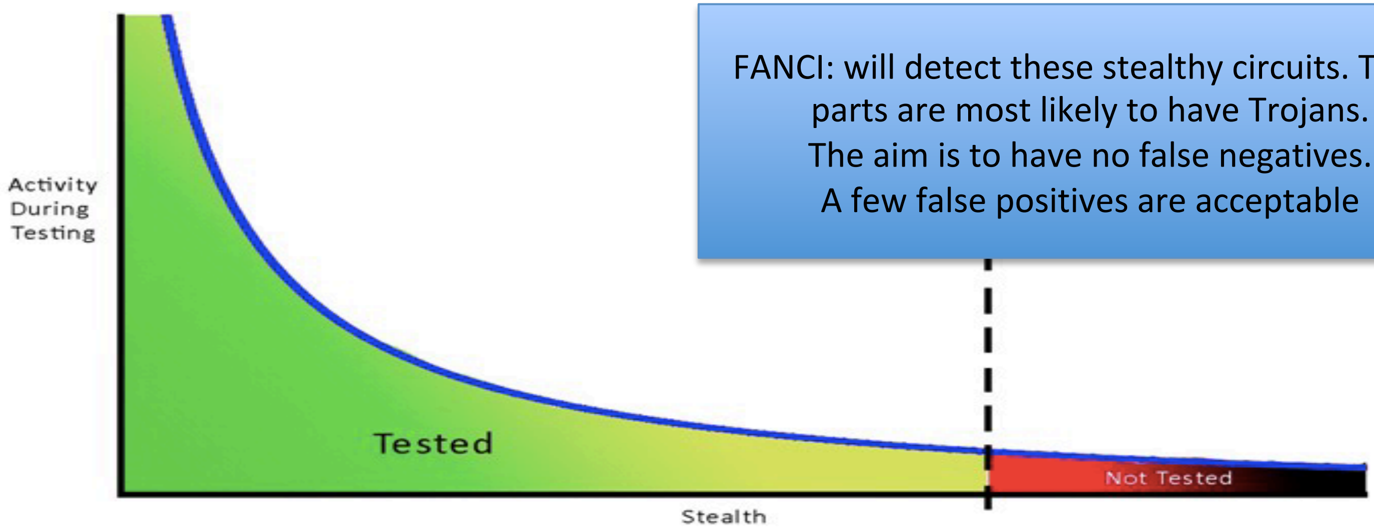
http://www.cs.columbia.edu/~simha/preprint_ccs13.pdf
(some of the following slides are borrowed from Adam Waksman's CCS talk)

Backdoors are Stealthy

- Small
  - Typically a few lines of code / area
- Stealth
  - Cannot be detected by regular testing methodologies (rare triggers)
  - Passive when not triggered

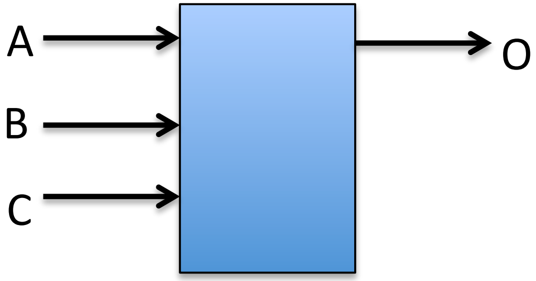
Unfortunately…

With so much of code it is highly likely that stealthy portions of the code are missed or not tested properly.



FANCI: will detect these stealthy circuits. These parts are most likely to have Trojans.
The aim is to have no false negatives.
A few false positives are acceptable

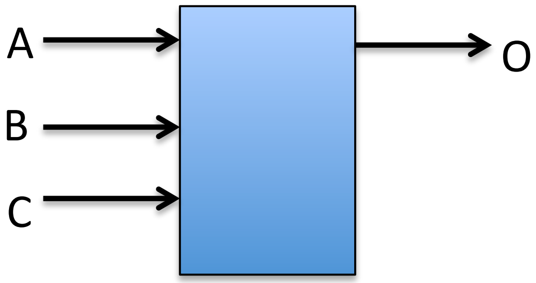
Control Values



By how much does an input influence the output O?

A	B	C	O
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

Control Values



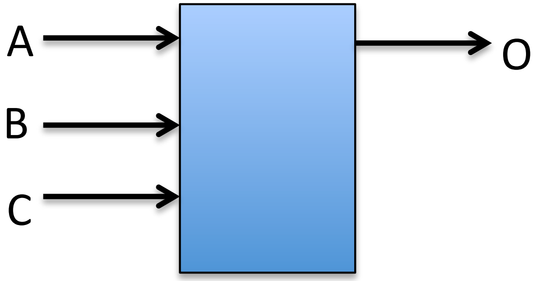
A	B	C	0
0	0	0	0
1	0	0	1
0	0	1	1
1	0	1	1
0	1	0	1
1	1	0	0
0	1	1	0
1	1	1	0

By how much does a input influence the output 0?

A : has a control of 0.5 on the output

(A matters in this function)

Control Values



A	B	C	0
0	0	0	0
1	0	0	0
0	0	1	1
1	0	1	1
0	1	0	0
1	1	0	0
0	1	1	0
1	1	1	0

By how much does a input influence the output 0?

A : has a control of 0 on the output

(A does not matter in this function)
(A is called unaffecting)

Control Values for a Trigger in a Trojan

```
if (addr == 0xdeadbeee) then{
    trigger = 1
}
```

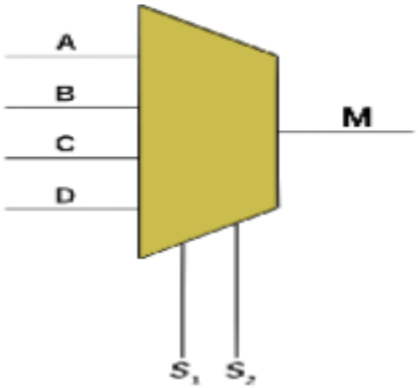
A31	A30		A2	A1	A0	trigger
0	0	…	0	0	0	0
0	0	…	0	0	1	0
0	0	…	0	1	0	0
0	0	…	0	1	1	0
:	:	:	:	:	:	
1	1		1	1	0	1
:	:	:	:	:	:	
1	1	1	1	1	1	0

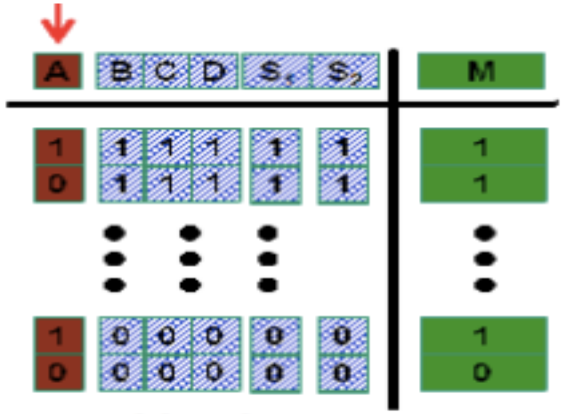
A31 has a control value $1/2^{16}$

Easier to hide a trojan when larger input sets are considered

A low chance of affecting the output
Lends itself to stealthiness →
easier to hide a malicious code

An Example of a Mux





Hold Other Inputs
Fixed in Pairs

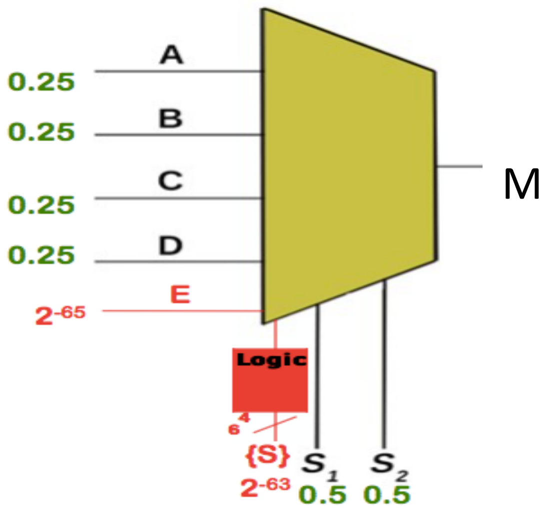
Control of A on M = 8/32 = 0.25

<A, B, C, D, S1, S2> = <0.25, 0.25, 0.25, 0.25, 0.5, 0.5>

No trojan present here (intutively):

* All mux inputs have a control value around mid range (not too close to 0)

An Example of a Malicious Mux



66 extra select lines which are only modify M when whey are set to a particular value

	A	B	C	D	E	S_1	S_2	{$S_{3\text{-}66}$}
M	0.25	0.25	0.25	0.25	2^{-65}	0.50	0.50	2^{-63}

The control values E and S3 to S66 are suspicious because they rarely influence the value of M.

Perfect for disguising malicious backdoors

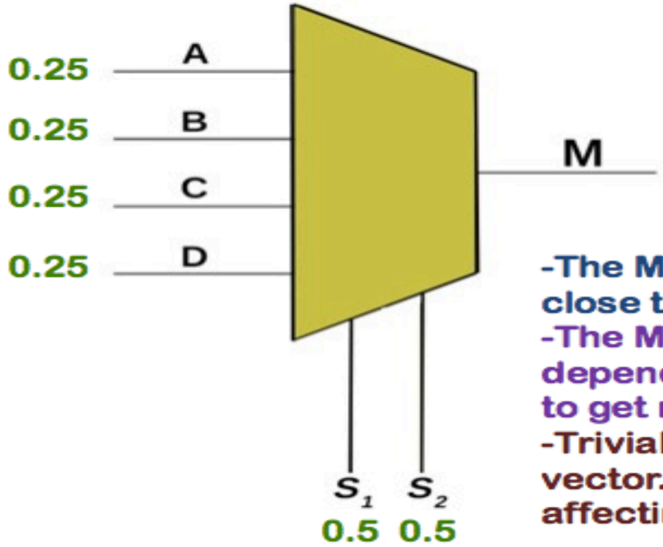
Just searching for MIN values is often not enough. Better metrics are needed.

Computing Stealth from Control

	A	B	C	D	S1	S2
M	0.25	0.25	0.25	0.25	0.50	0.50

We use three different heuristics for evaluation. Mean, Median and Triviality.

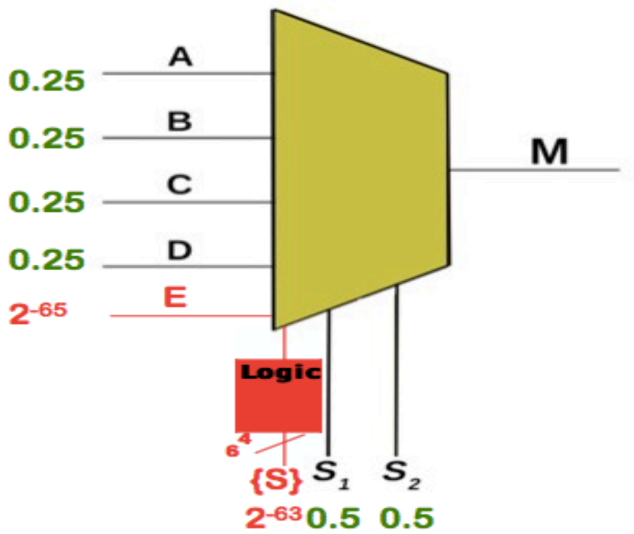
Mean(M) = (2.0 / 6) = <u>0.33</u>
Median(M) = <u>0.25</u>
Triviality(M) = <u>0.50</u>



-The Median in the context of backdoor triggers is often close to zero when low or unaffecting wires are present.
-The Mean is sensitive to outliers. If there are few dependencies, and one of them is unaffecting, it is likely to get noticed, when compared to the control value.
-Triviality is a weighted average of the values in the vector. Weighted by how often they are the only value affecting the output. If it is 0 or 1 it is trivial.

Computing Stealth from Control

	A	B	C	D	E	S1	S2	{S3-66}
M	0.25	0.25	0.25	0.25	2^{-65}	0.50	0.50	2^{-63}



Mean(M) = (2.0 / 71) = 0.03

Median(M) = 2^{-63}

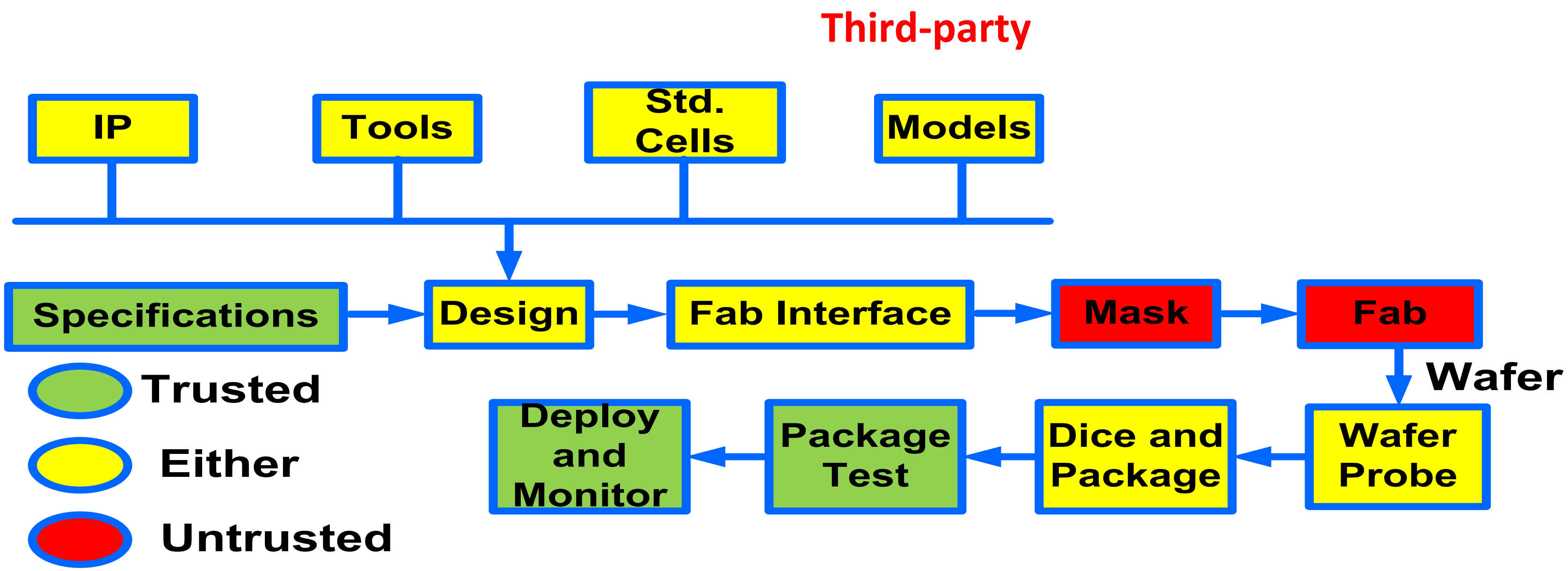
Triviality(M) = 0.50

FANCI: The Complete Algorithm

Algorithm 1 Flag Suspicious Wires in a Design

```
for all modules m do
    for all gates g in m do
        for all output wires w of g do
            T ← TruthTable(FanInTree(w))
            V ← Empty vector of control values
            for all columns c in T do
                Compute control of c (Section 3.2)
                Add control(c) to vector V
            end for
            Compute heuristics for V (Section 3.3)
            Denote w as suspicious or not suspicious
        end for
    end for
end for
```

IC Life Cycle
(The Fab)



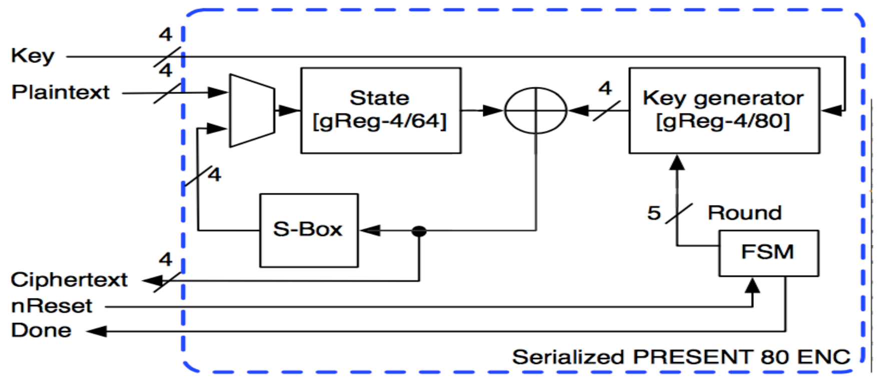
*http://www.darpa.mil/MTO/solicitations/baa07-24/index.html

Detecting Trojans in ICs

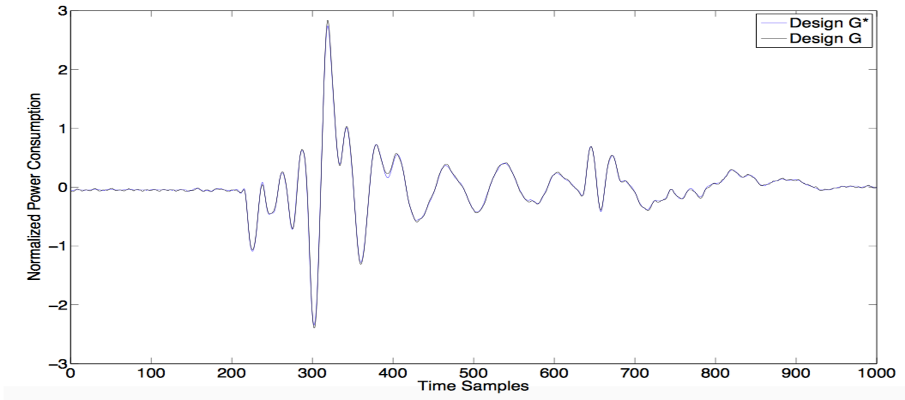
- Optical Inspection based techniques
  Scanning Optical Microscopy (SOM),
  Scanning Electron Microscopy (SEM),
  and pico-second imaging circuit analysis (PICA)
  – Drawbacks: Cost and Time!
- Testing techniques
  – Not a very powerful technique
- Side channel based techniques
  – Non intrusive technique
  – Compare side-channels with a golden model

A Survey on Hardware Trojan Detection Techniques
http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7169073

Side Channel Based Trojan Detection



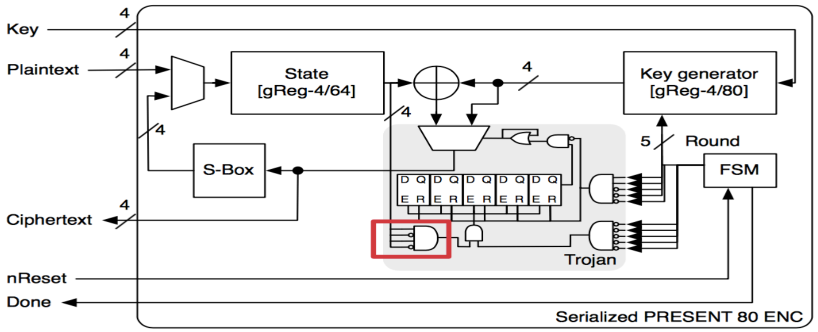
Lightweight PRESENT Implementation

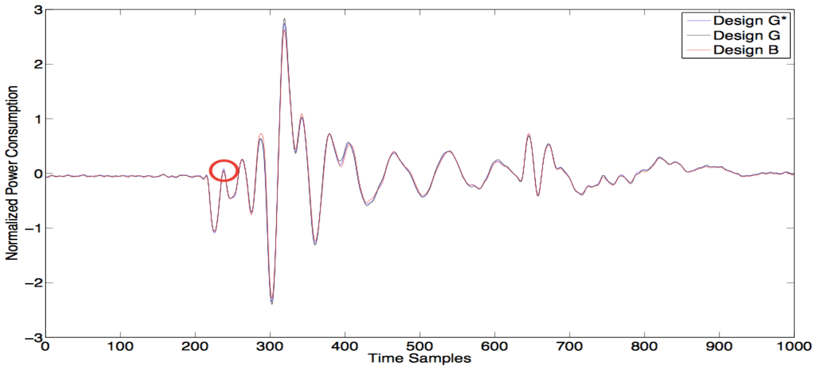


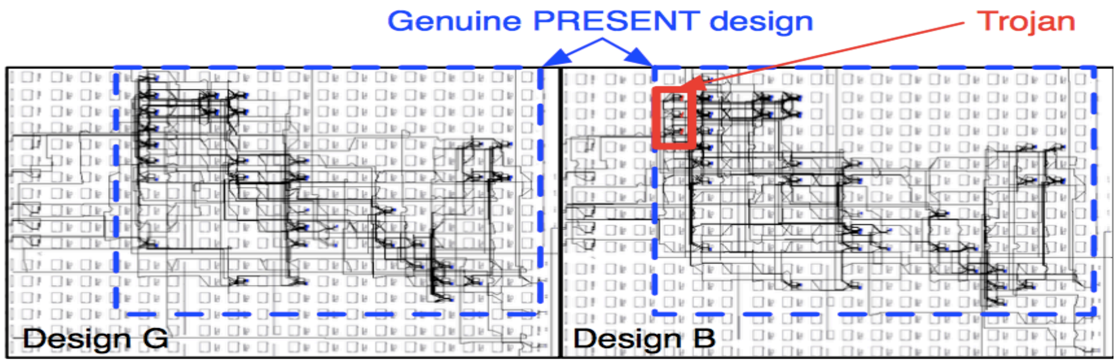
Power Traces

Hardware trojan design and detection: a practical evaluation
https://dl.acm.org/citation.cfm?id=2527318

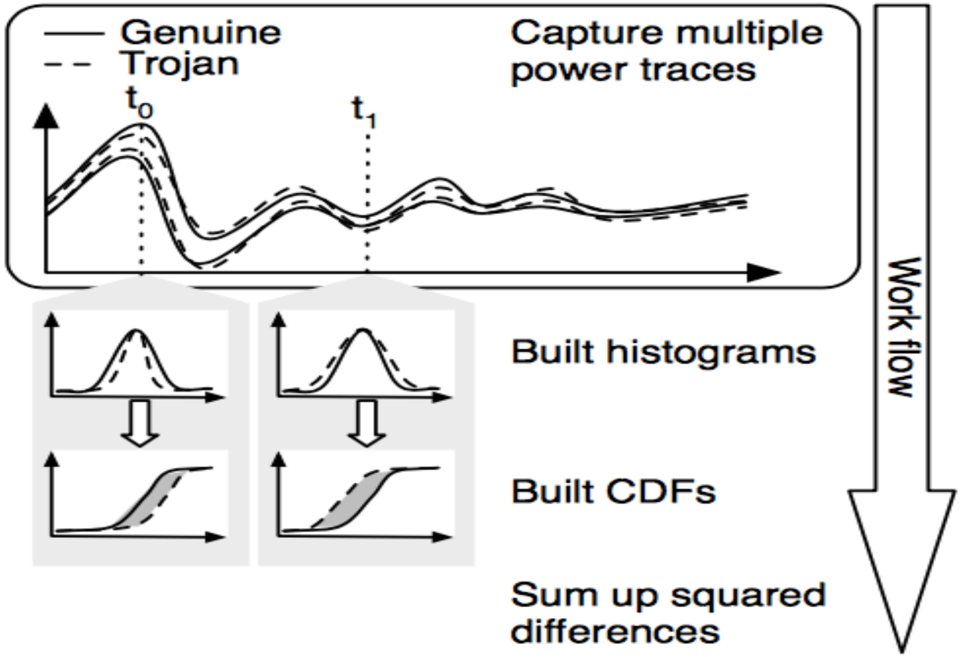
Side Channel Based Trojan Detection (IC with Trojan)







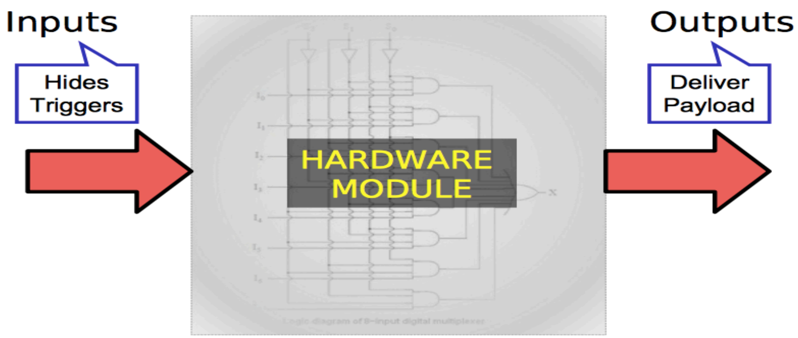
Difference of Distributions



Hardware Trojan Prevention

(If you can’t detect then prevent)

Backdoor = Trigger + Payload

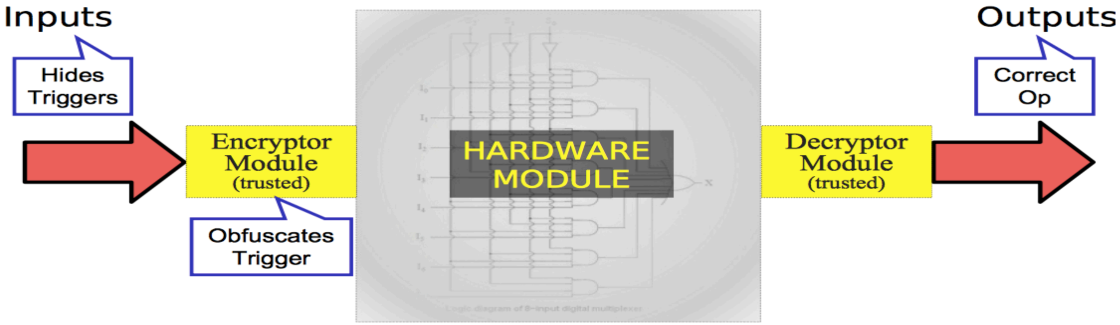


[Silencing Hardware Backdoors](www.cs.columbia.edu/~simha/preprint_oakland11.pdf)

www.cs.columbia.edu/~simha/preprint_oakland11.pdf

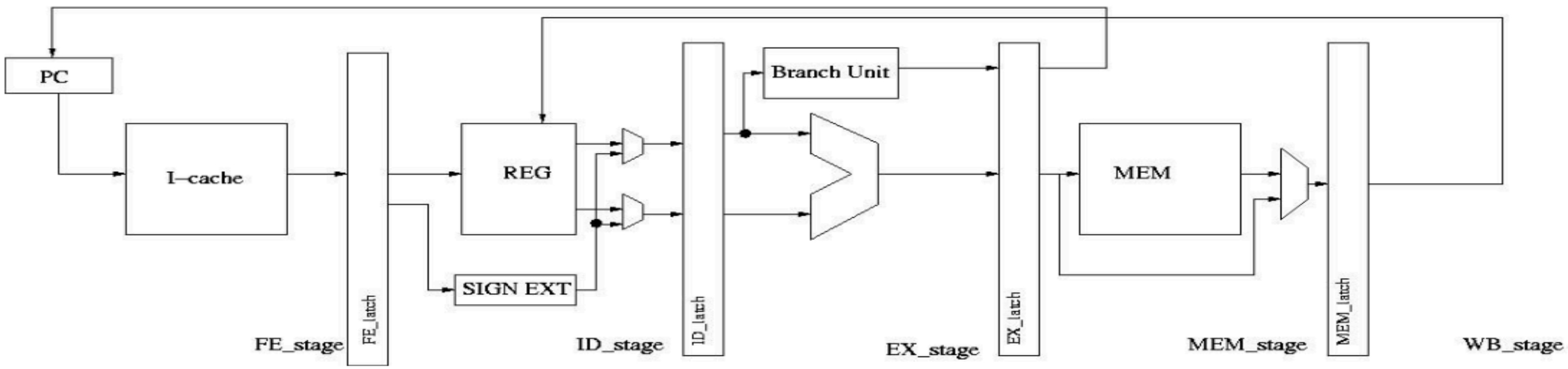
Slides taken from Adam Waksman’s Oakland talk

Hardware Trojan Prevention



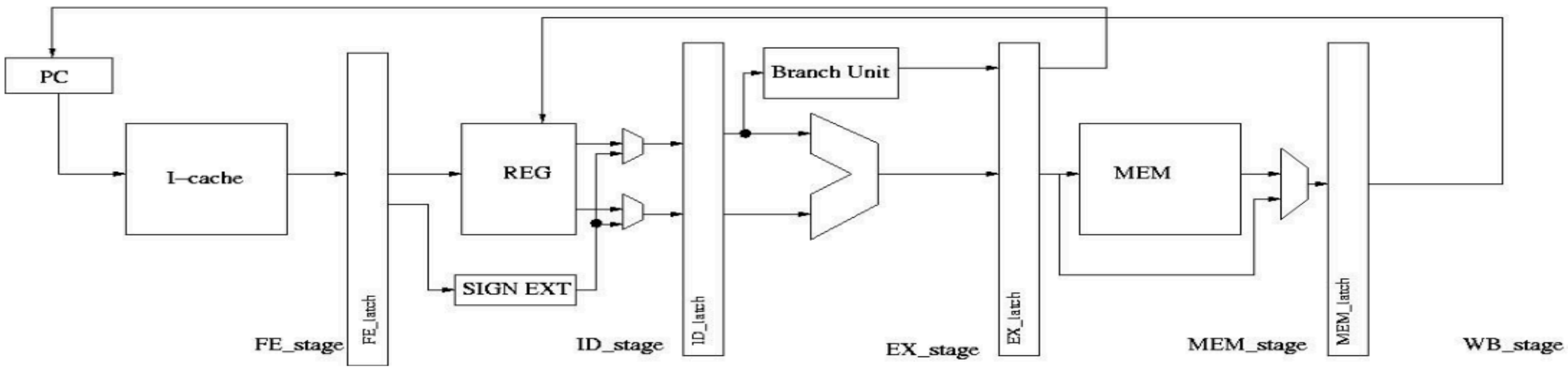
Ensure that a hardware Trojan is never delivered the correct Trigger

Example (A 5 stage processor)



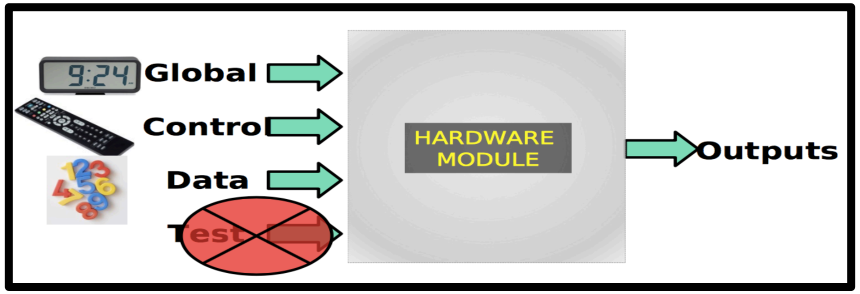
- A design is a connected set of modules
  - Modules connect to each other through interfaces
- In the picture above, each box is a module

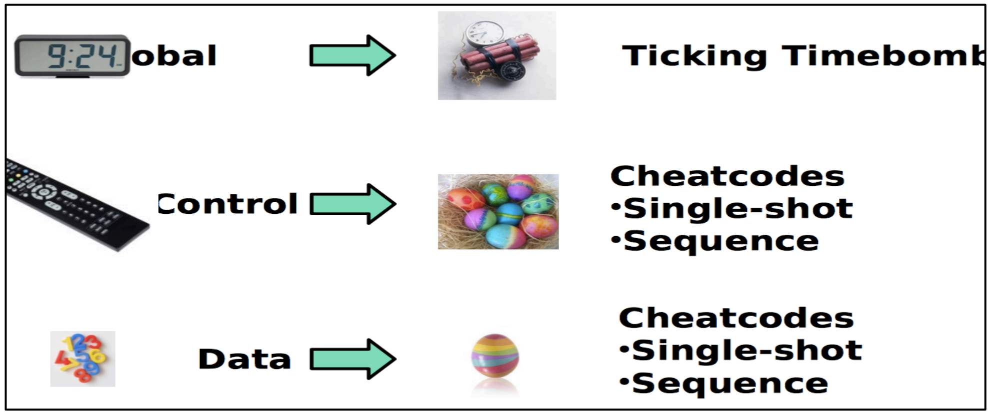
Example (A 5 stage processor)



- A design is a connected set of modules
  - Modules connect to each other through interfaces
- In the picture above, each box is a module

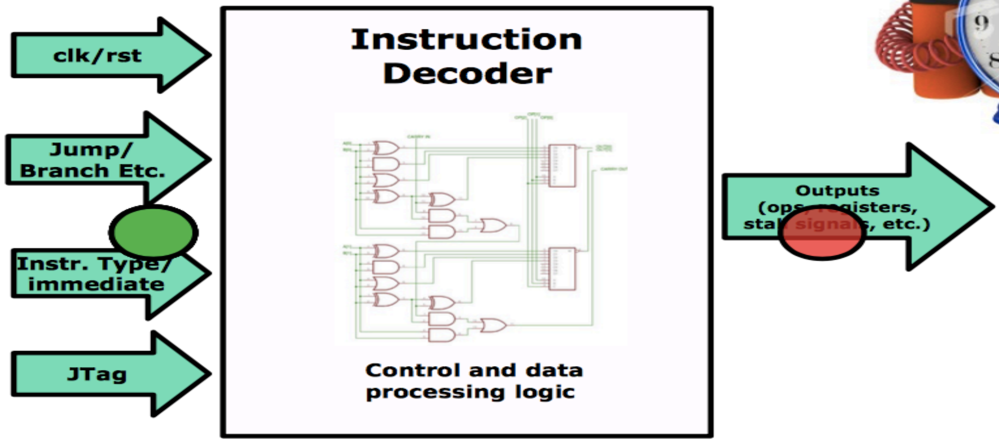
Types of Trojans



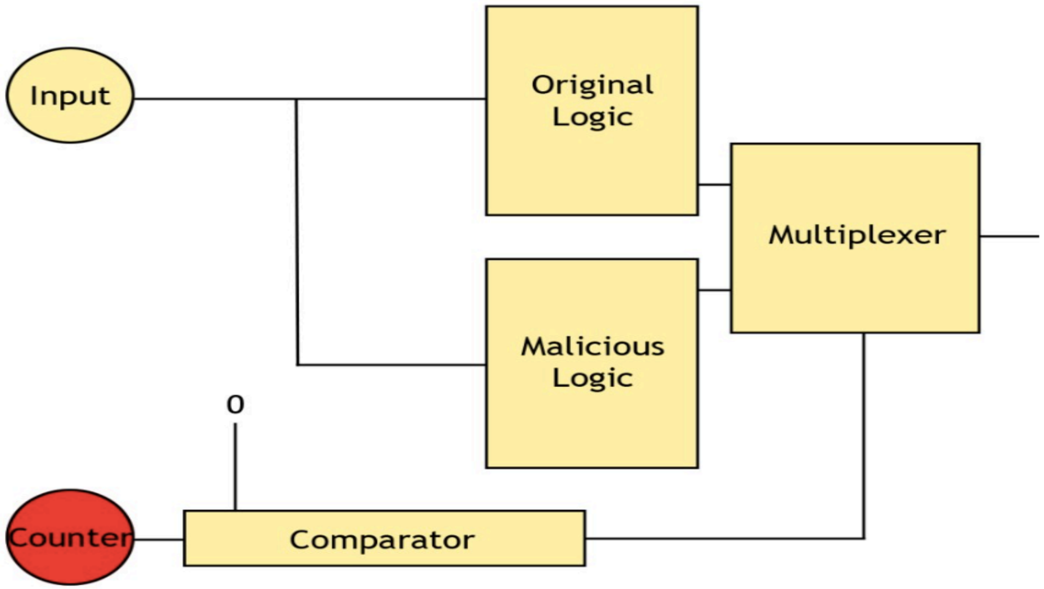


Ticking Timebomb

- After a fixed time, functionality changes

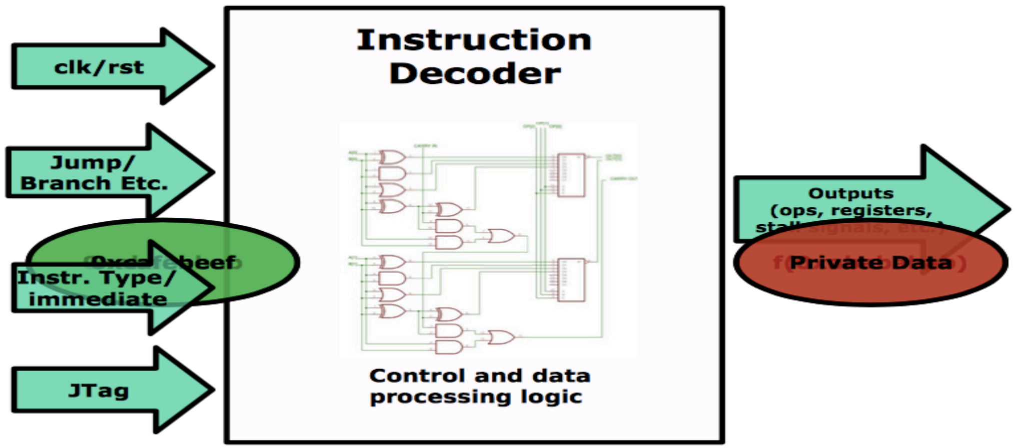


Ticking Timebomb



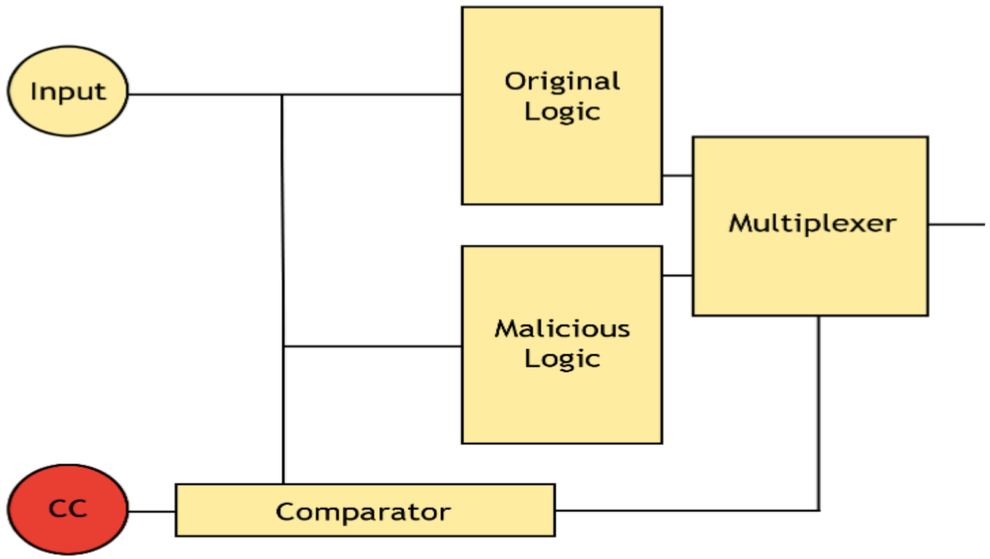
Cheat Codes

- A special value turns on malicious functionality
  - Example: 0xcafebeef



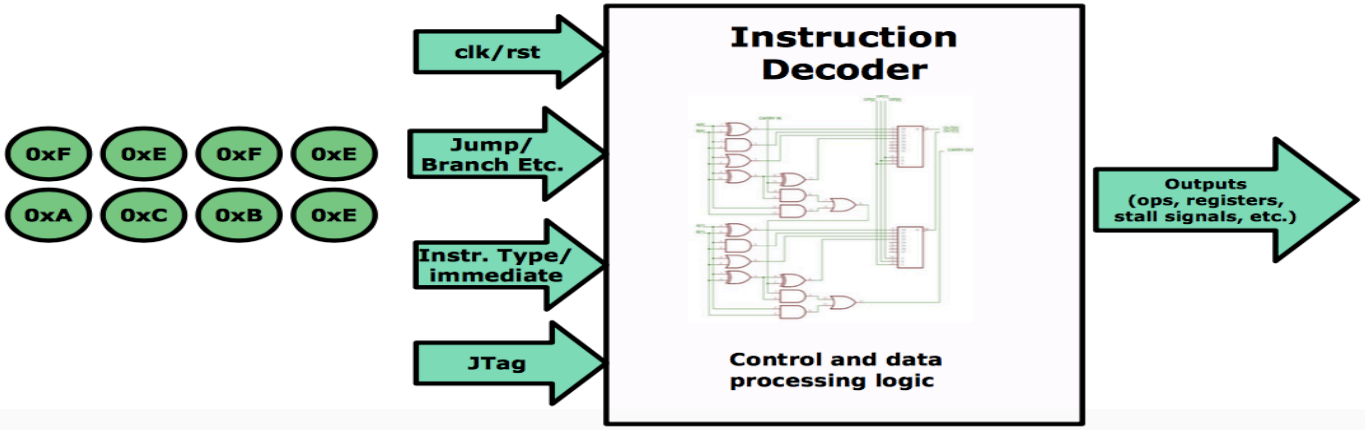
Cheat Codes

- Example: 0xcafebeef

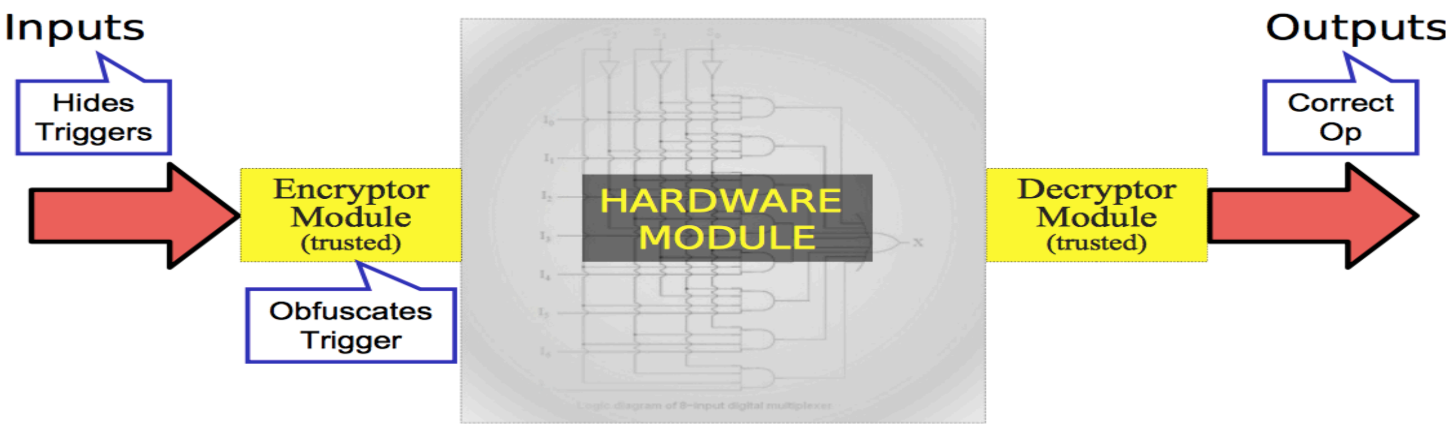


Sequence Cheat Codes

- A set of bits, events, or signals cause malicious functionality to turn on
  - Example: c, a, f, e, b, e, e, f



Hardware Trojan Silencing (with Obfuscation)



Silencing Ticking Timebombs

- Power Resets : flush pipeline, write current IP and registers to memory, save branch history targets

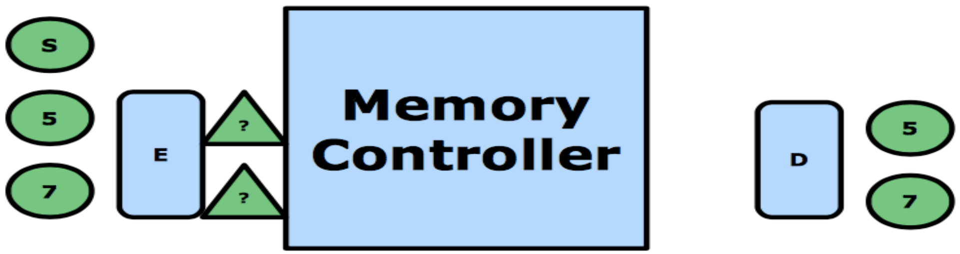
•Power to modules is reset periodically
- Time period = N – K cycles
- N = Validation epoch
- K = Time to restart module operation

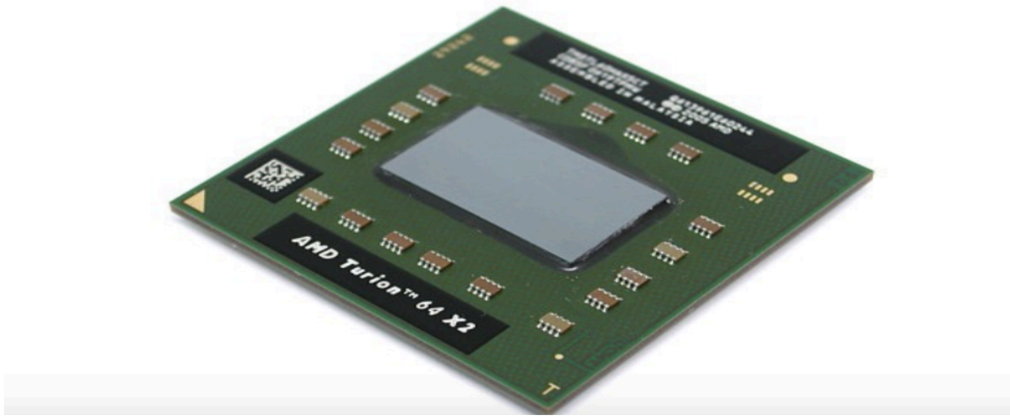
•Forward progress guarantee
- Architectural state must be saved and restored
- Microarchitectural state can be discarded (low cost)
  - e.g., branch predictors, pipeline state etc.,

Silencing Ticking Timebombs

- Can trigger be stored to architectural state and restored later
  - No. Unit validation tests prevent this
  - Reason for trusting validation epoch
    - Large validation teams
    - Organized hierarchically
- Can triggers be stored in non-volatile state internal to the unit?
  - Eg. Malware configures a hidden non-volatile memory
- Unmaskable Interrupts?
  - Use a FIFO to store unmaskable interrupts
- Performance Counters are hidden time bombs

Data Obfuscation

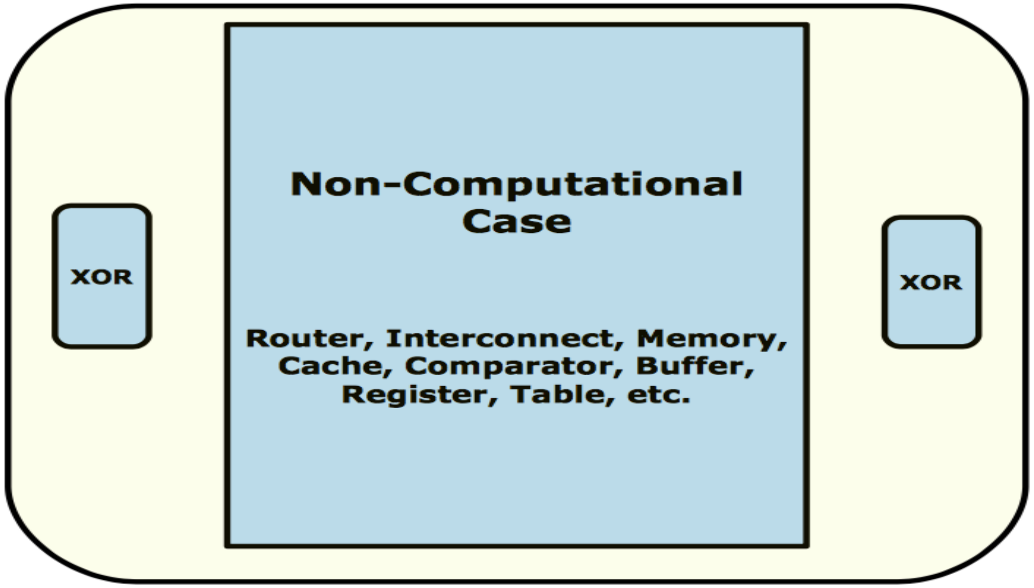




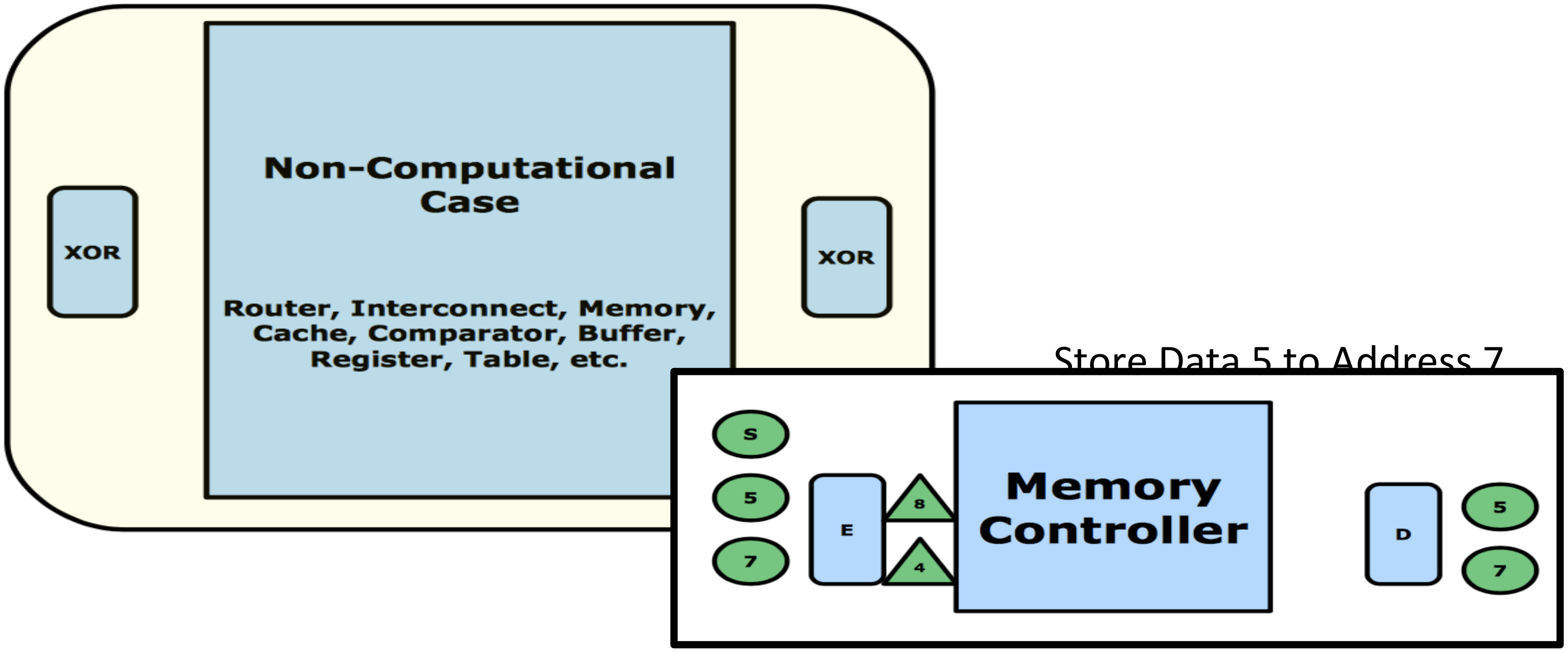
Homomorphic Encryption
(Gentry 2009)

Ideal solution
But practical hurdles

Data Obfuscation

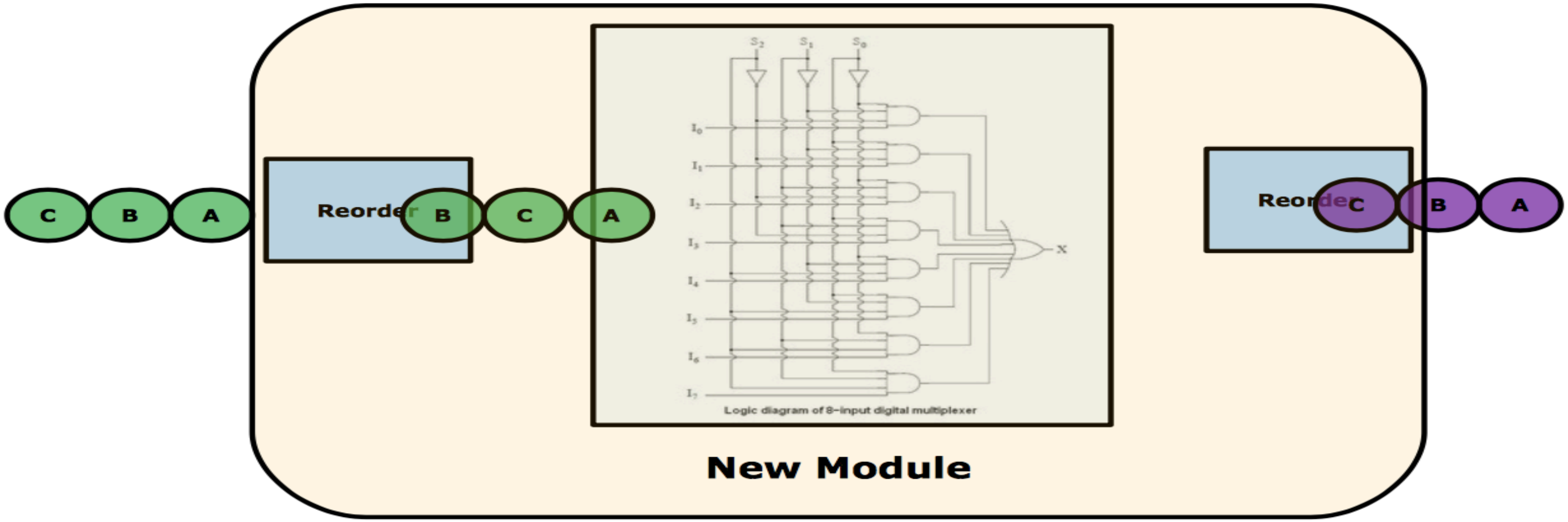


Data Obfuscation



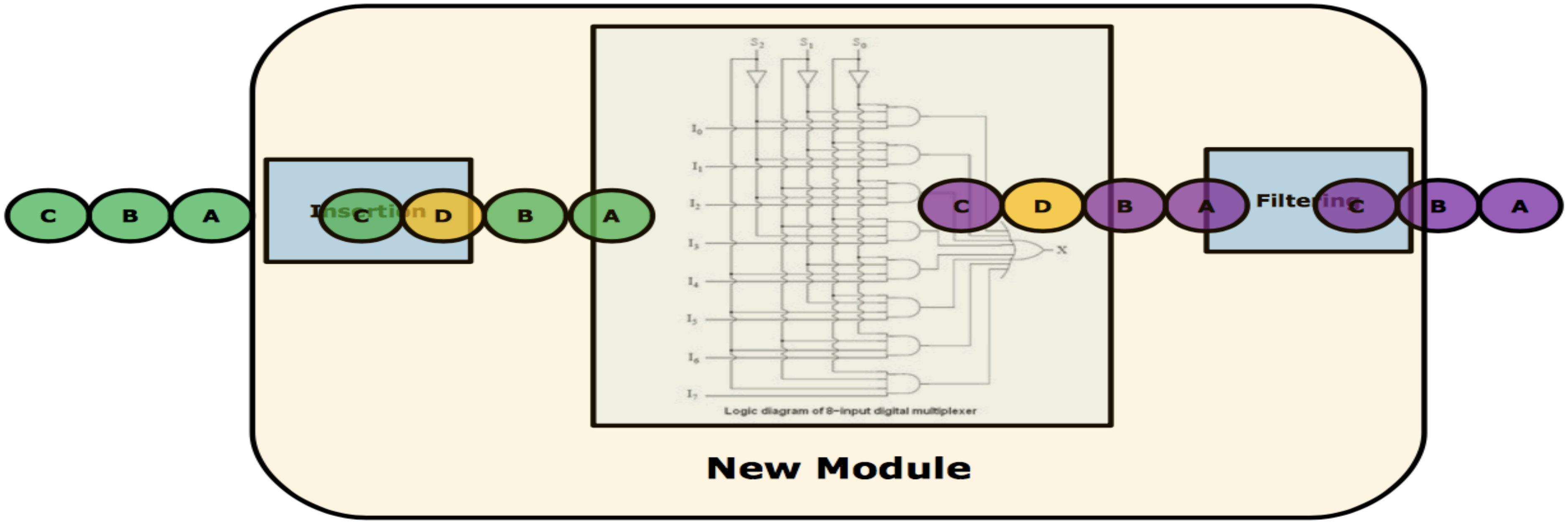
Data Obfuscation (Computational Case)

Sequence Breaking (Reordering)



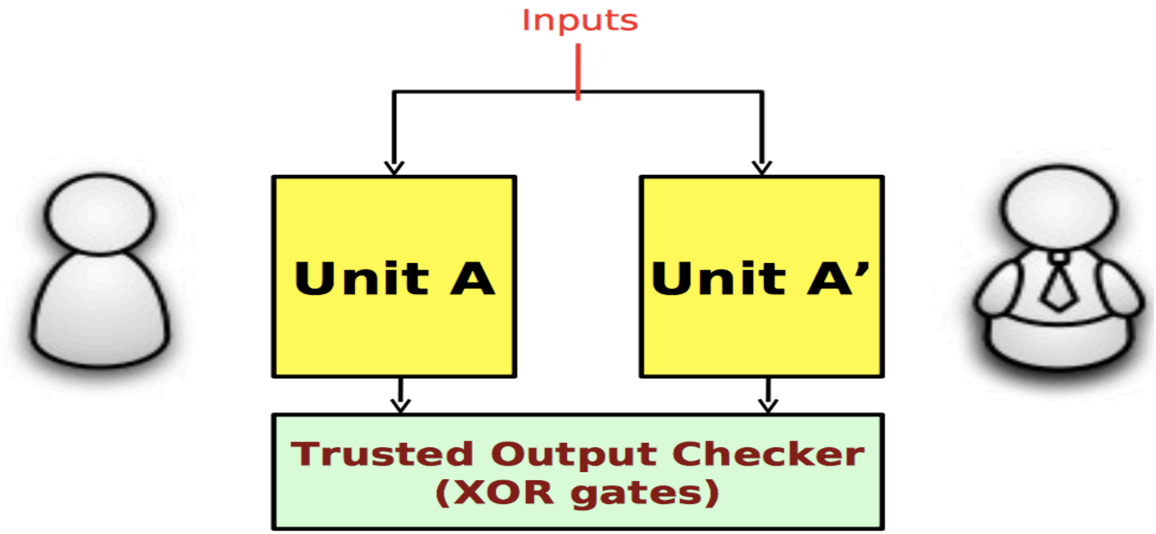
Ensure functionality is maintained

Sequence Breaking (Inserting events)



Insert arbitrary events when reordering is difficult

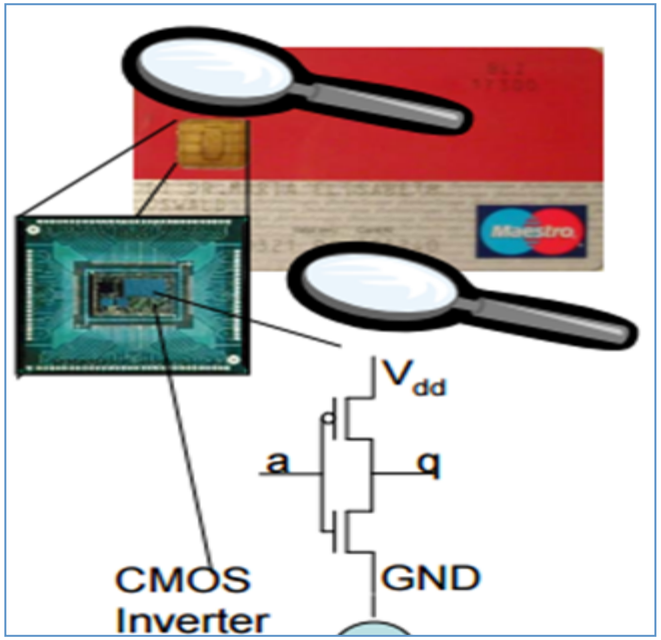
Catch All (Duplication)



Expensive:
Non-recurring : design; verification costs due to duplication
Recurring : Power and energy costs

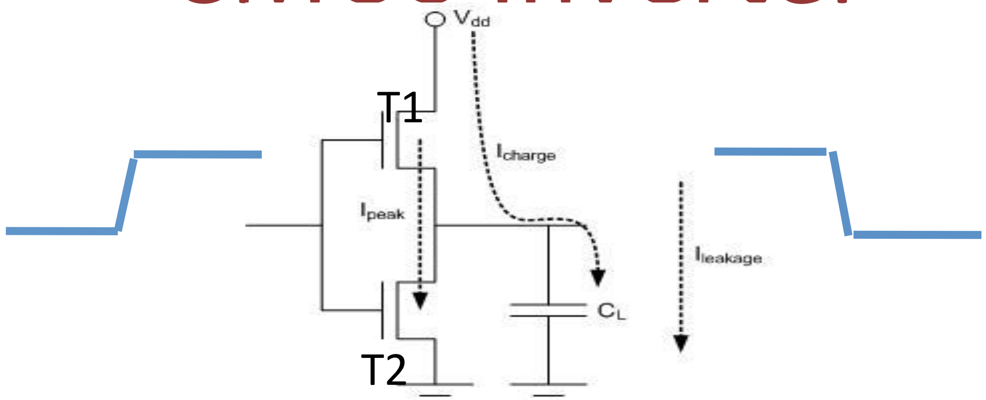
Power Analysis

CMOS Technology



- Almost every digital device is built using CMOS technology.
- CMOS – complimentary metal oxide semiconductor

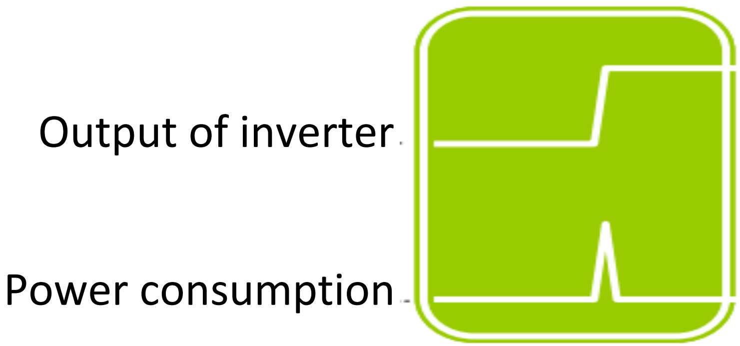
CMOS Inverter



- When the input switches from 0 → 1, Transistor T1 turns on and T2 turns off. Capcitor CL gets charged.
- When the input switchs from 1→0, transitor T1 is turned off and T2 turns on. Capacitor CL discharges.

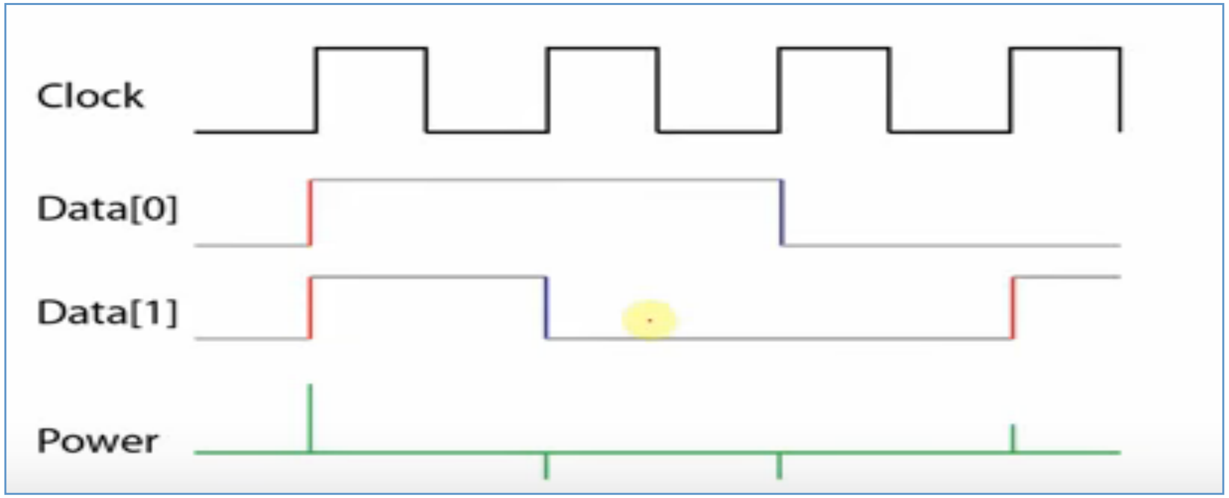
Power Consumption of a CMOS Inverter

- Power is consumed when CL charges or discharges (i.e. there is a transition in the output from 0 → 1 or 1→0)
- Using an oscilloscope we can measure the power to determine when the inverter output changes state



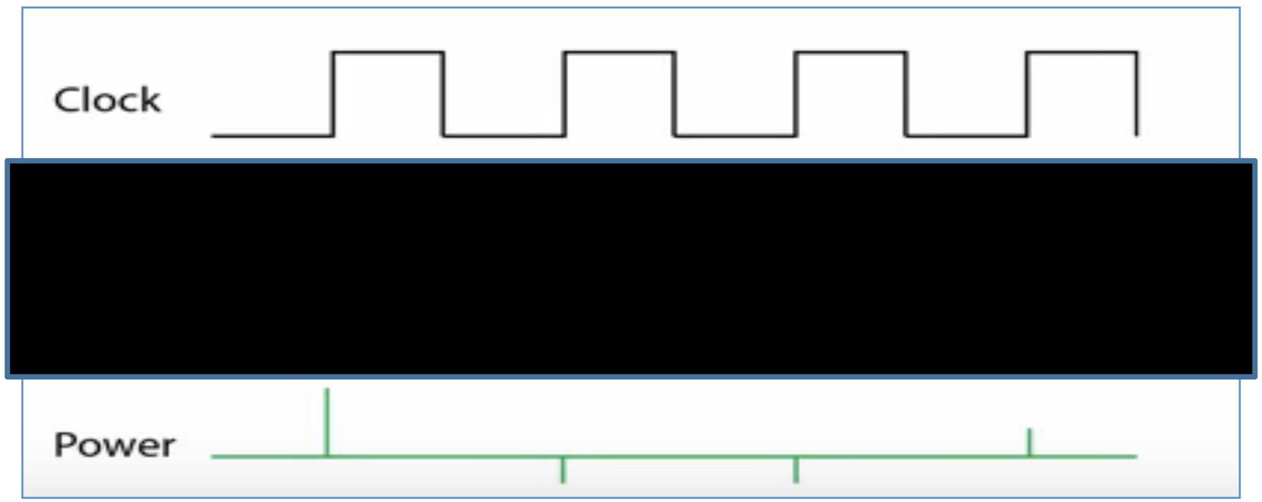
Synchronous Digital Circuits

- Most electronic equipment use a clock as reference
- All state transitions are done with respect to this clock
  - Power consumption is therefore at clock edges



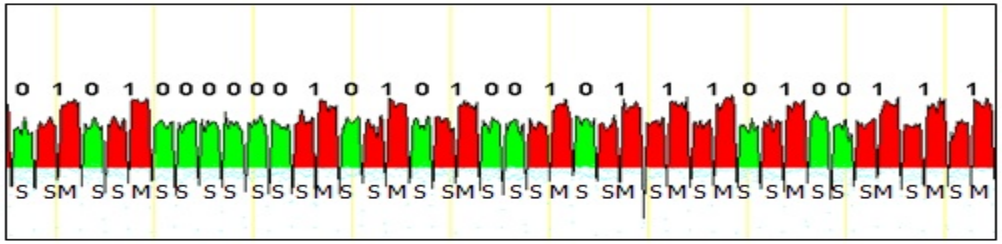
Essence of Power Analysis

- We don’t know what is happening inside the device, but we know the power consumption
- Can we deduce secret information from the power consumption



The Types of Power Analysis

- SPA : Simple Power Analysis

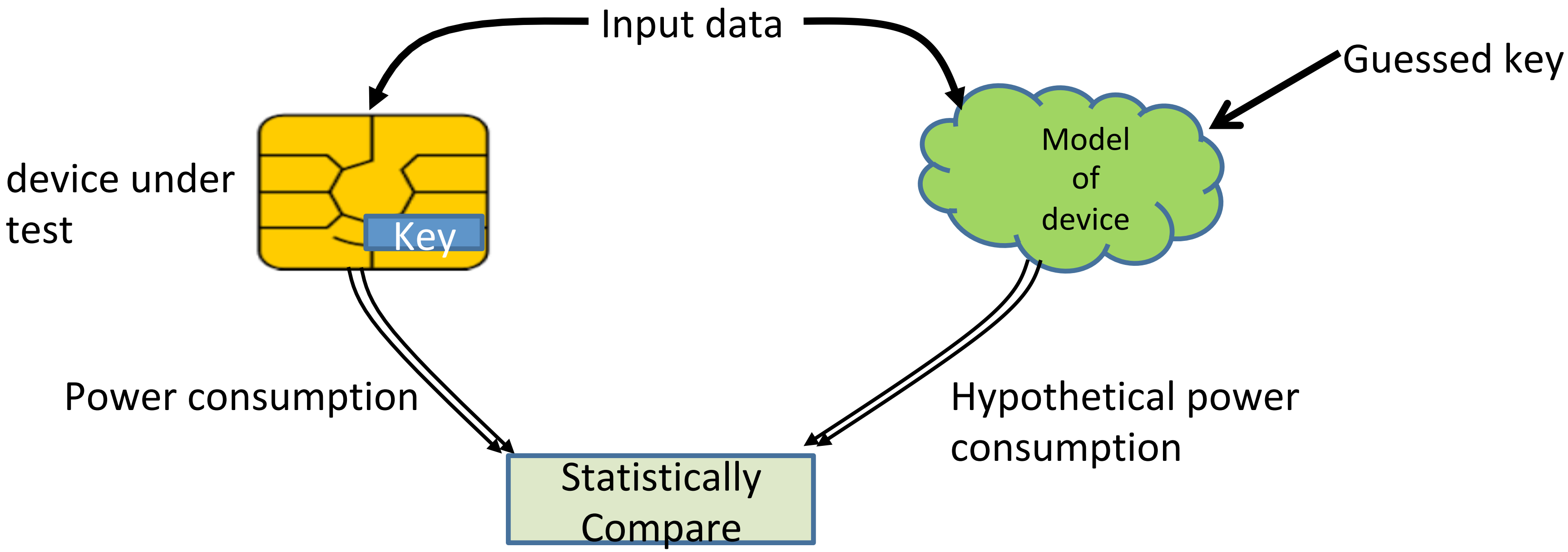


- DPA : Differential Power Analysis

  Requires more strategy and statistics to glean secret information

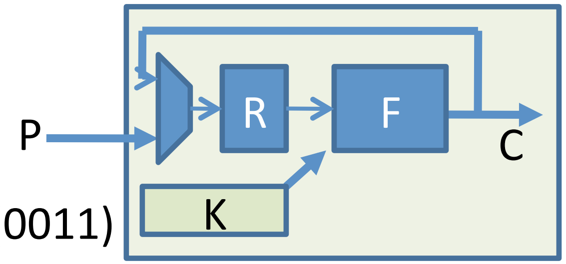
- Template based attacks

Differential Power Analysis (as a glance)



Hypothetical Power Consumption

- CMOS circuits follow the Hamming weight and Hamming distance power models
- Hamming Distance Model
  - Consider transitions of register R



(1011) → (1101) → (1001) → (0010) → (0011)

#toggles **3** **1** **3** **1**

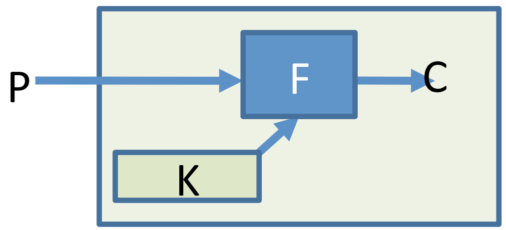
- Hamming Weight Model

(1011) → (1101) → (1001) → (0010) → (0011)

#toggles **3** **2** **1** **3**

The Hamming weight model will work, when R is precharged to either 0 or 1

A Small Example



Device

P	K	C
0000	1010	1010
0001	1010	1011
0010	1010	1000
0011	1010	1001
0100	1010	1110
0101	1010	1111
..	…	…

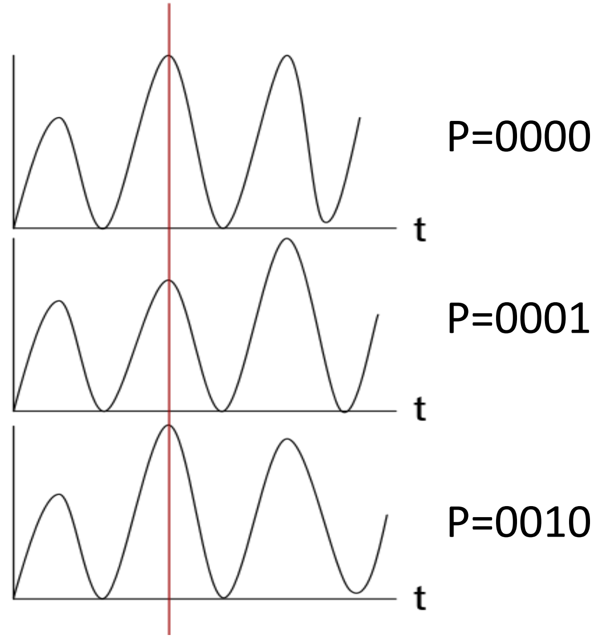
Mallory has control of this device.
-- She can monitor its power consumption
-- She can feed inputs **P**
-- She even knows what operations goes on inside.

The things she doesn't know is K and C
Her aim is to obtain the secret key **K**

DPA Attack

P	K_{guess}	C	Hypothetical Power	Real Power Measured
0000	1111	1111	4	
0001	1111	1110	3	
0010	1111	1101	3	
0011	1111	1100	2	
0100	1111	1011	3	
0101	1111	1010	2	
⋮	⋮	⋮	⋮	⋮

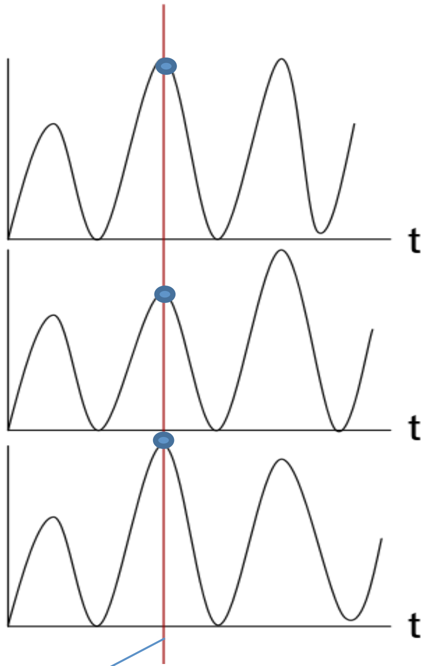
note that this is a waveform which changes w.r.t time



C here is computed wrt to the guessed key i.e. $C = F(P, K_{guess})$

DPA : What we mean by correlation

Hypothetical Power
4
3
3



correlate

These waveforms are discrete,
they have several points

Perform correlation of hypothetical
Power wrt each point in the waveforms

Consider only the maximum correlation

DPA : A small example

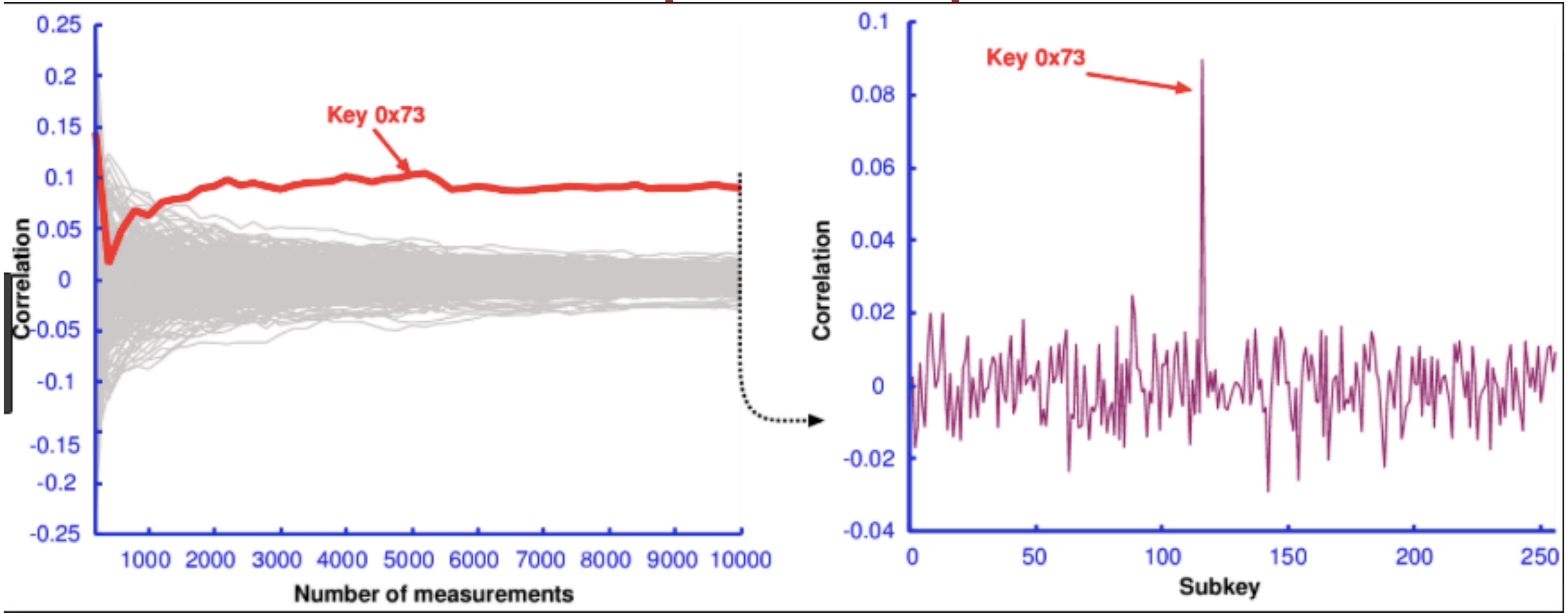
P	K_{guess}	C	Hypothetical Power	Real Power Measured
0000	1101	1101	3	xx
0001	1101	1100	2	xx
0010	1101	1111	4	xx
0011	1101	1110	3	xx
0100	1101	1001	2	xx
0101	1101	1000	1	xx
⋮	⋮	⋮	⋮	⋮

correlate

Find maximum correlation

ρ_{15} ρ_{14} ρ_{13} ρ_{12} ρ_{11} ρ_{10}

Sample Output



https://iis-people.ee.ethz.ch/~kgf/acacia/acacia.html

Statistical Comparison

- Correlation :

  Provides a value between -1 and +1. A value closer to the signifies linear dependence between the hypothetical power and the real power consumption

$$\rho_{X,Y} = \frac{\mathrm{E}[(X-\mu_X)(Y-\mu_Y)]}{\sigma_X \sigma_Y}$$

- Mutual Information

  Quantifies mutual dependence between hypothetical power and real power consumption

$$I(X;Y) = \sum_{y \in Y} \sum_{x \in X} p(x,y) \log\left(\frac{p(x,y)}{p(x)\,p(y)}\right)$$

Statistical Comparison

- Bayes Analysis

  What is the probability of a hypothesis given a specific leakage

  Pr[Hypothesis | Leakage]

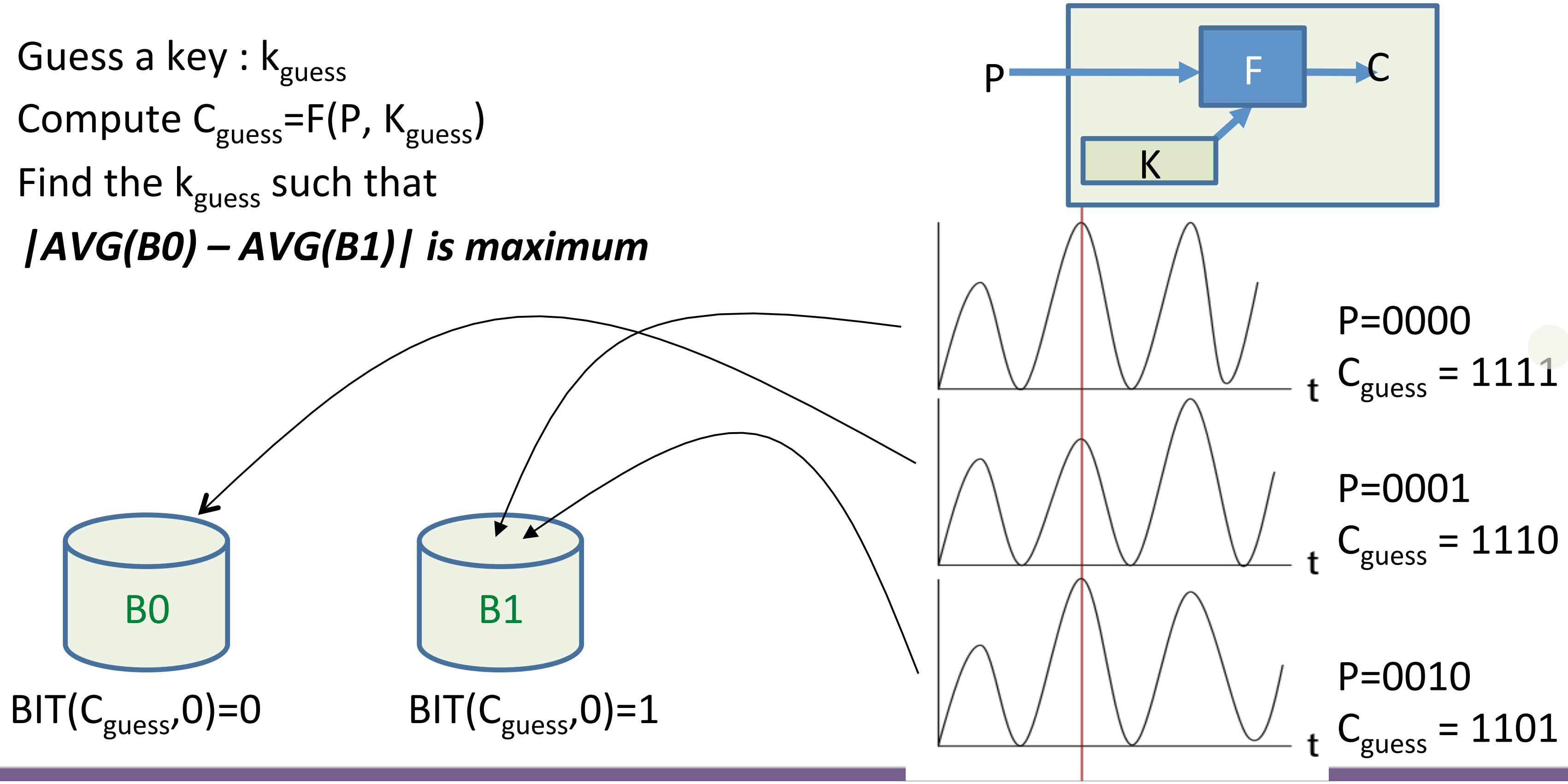
- Difference of Means

  next…

Difference of Means

- Guess a key : k_{guess}
- Compute $C_{guess}=F(P, K_{guess})$
- Find the k_{guess} such that

  |AVG(B0) – AVG(B1)| is maximum



Preventing DPA

- By hardware means
  - Differential logic
- By Implementation
  - Masking
- By Algorithm
  - DPA resistant ciphers (DRECON)
  - Rekeying