Hardware & Embedded Security

Prof Daniele Rossi


Via G. Caruso 16, room B-1-03


daniele.rossi1@unipi.it


050 221 7611

1

Physically Unclonable Functions

Lecture 6 - DR

2

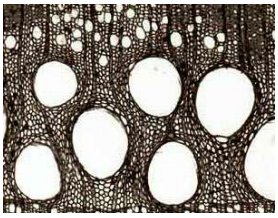
Brief outline

- **Design and evaluation principles of PUF**
- Quality metrics for PUF evaluation
- PUFs generating/storing cryptographic keys
- PUFs for entity authentication protocols
- PUFs for tamper resistant hardware
- PUFs for hardware metering

Examples of possible applications

3

Physical Disorder Characteristics






5/1/2025

4

This is one of the two fundamental CS HW primitives: the PUFs.
What are them? What's the principle behind them? The physical disorder behind the real world. Even two identical objects are different if you go at low level enough. In reality there are no two equal things.

- Disorder is omnipresent and makes things hard (impossible) to clone (make two things exactly equal).
  - This disorder is hard to characterize and of course is hard to simulate.

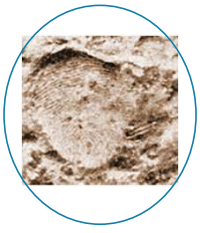
We can use this disorder to our advantage: our disorder is PROCESS VARIATION

Physical Disorder Characteristics

1. Omnipresent
2. Hard to Clone
3. Hard to Fully Characterize
4. Hard to Simulate on a Turing Machine

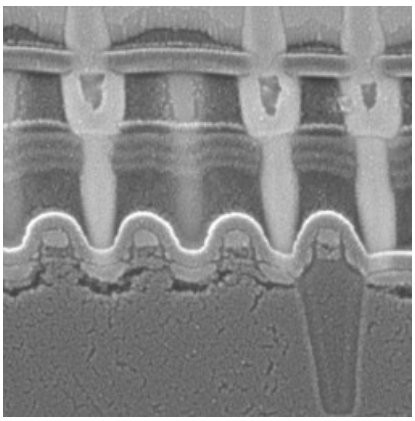
5

Babylonians' Clay Fingerprints



6

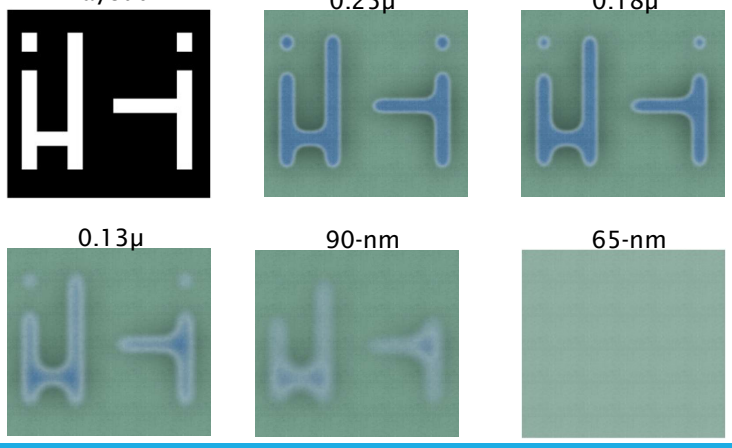
Physical Disorder of Silicon Chips



- Cross-section of the iPad mini's A5 processor. Picture courtesy of Chipworks

7

Limitation of Chip Fabrication*

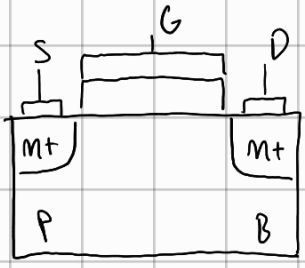


From masks I expect to find a right geometry but as feach scales, we have more and more problems

*Picture courtesy of Chipworks

8

So, structures that are messy at low scale can be used to our advantage. For process variation we have intra-die (across same wafer we have parameters variations) or inter-die variation: within same die electric transistors characteristics vary and are spatially correlated (see psc slide 9). What are those electric characteristics?



We mean channel length, oxide width, doping concentration, threshold voltage etc.

To give an idea of standard deviation of threshold voltage: it increases as we work with smaller technologies (250nm → 45nm; nm is channel length).
So we can exploit them to realize PUFs in a way that identifies a fingerprint that characterizes our chip.

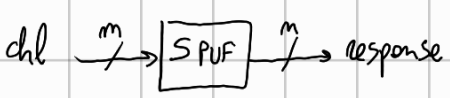
So, SILICON BASED PUF (because we work w/ silicon): because of process variation, behavior of a chip is unique. This can be used to uniquely id. a device.
If I have two IC to which I apply a stimulus (challenge), they will produce two different responses even if they are functionally the same. Responses are our digital fingerprints. This challenge-response can be used to authenticate a device.

- Main physical causes: oxide, doping and feature size.

PUFs can be divided between Weak and Strong PUFs.

- Weak Pufs don't have challenges to be applied, or just one.
  ↳ "A puf is weak whe number of challenge response pairs is limited: if n increase complexity of chl complexity increases linearly".
  - Applications are for identity gen, RNG seed generation, etc.

- Strong PUFs may work with many challenge and response pairs. An example is Arbiter PUF.



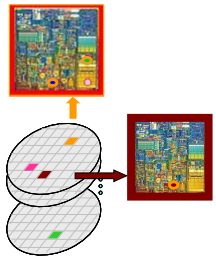
If pairs chl-esp are a lot it should be impossible to find a model that, looking

a/ chl - rsp we have, predicts response to another challenge.

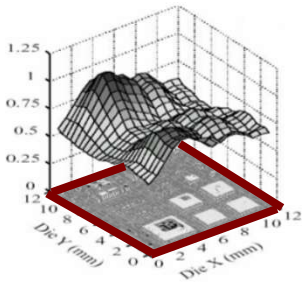
Principle of a PUF; an example: JMP 16

Process Variation Classification

- Inter-die vs intra-die variation
  - **Inter-die variation**: same devices at different dies are manufactured differently
  - **Intra-die** (spatial) variation: same devices at different locations of the same die are manufactured differently



Inter-die variation



Intra-die variation

9

9

Limitation of Chip Fabrication

- With technology scaling, the parameters of devices on the same die show **increasing intra-die variations**, thereby exhibiting different characteristics. For example, the table below displays the evaluation of the typical transistor's threshold voltage standard deviation $\sigma_{V_{th}}$ normalized by the threshold voltage V_{th} for several technologies.

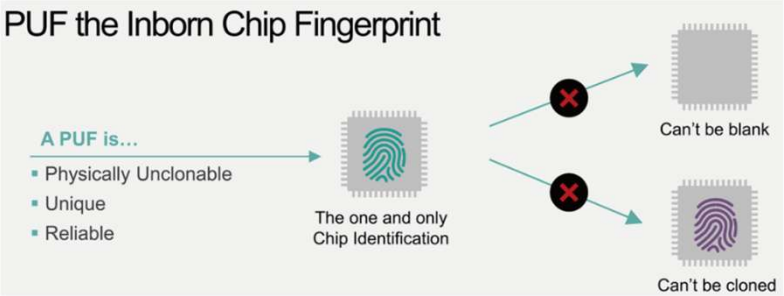
Table. Intra-die variability vs. CMOS technology node

Technology Node	250nm	180nm	130nm	90nm	65nm	45nm
$\frac{\sigma_{V_{th}}}{V_{th}}$	4.7	5.8	8.2	9.3	10.7	16

10

Silicon-Based Physical Unclonable Function

- Physically unclonable functions (PUFs) are used in hardware security primarily for **chip identification** and **authentication**.



- Every chip ever produced has natural process variations that make it physically **unique** from one another, even for dies from the same wafer.

11

Silicon-Based Physical Unclonable Function

- PUFs exploit IC manufacturing process variations to generate a **unique key** → **unique chip's fingerprints**.
- Map a set of challenges to a set of responses

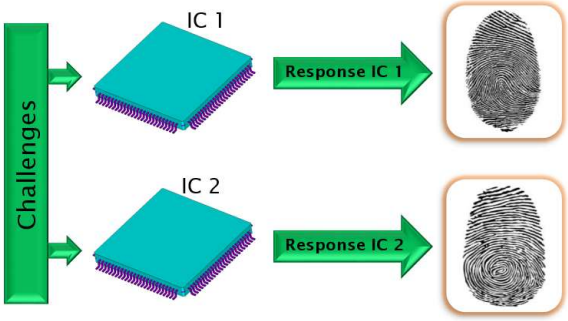
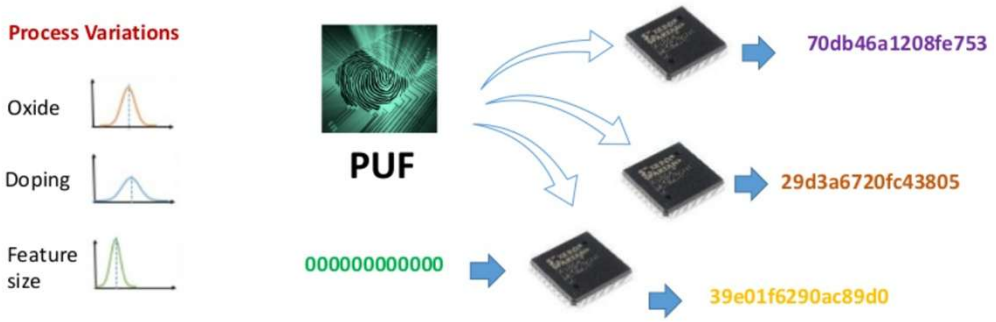


Figure: Unique Response Generation

12

Silicon-Based Physical Unclonable Function

A PUF (Physical Unclonable Function) is a digital circuit that uses manufacturing process variations to generate a unique digital fingerprint.



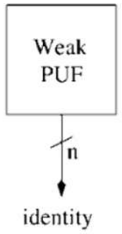
No two chips should give the same response when supplied with the same challenge.

13

PUF Classification: Weak vs Strong

Weak PUF

- Typically have no (or one fixed) challenge
  - e.g. SRAM PUF, Butterfly PUF.
- Assumed an attacker cannot access the responses of "Weak" PUFs as one or few CRPs could be used to build a model of the security system
- Applications include:
  - Identity generation
  - RNG seed
  - Non volatile key storage
  - Hardware root of trust
  - Anti-Cloning



14

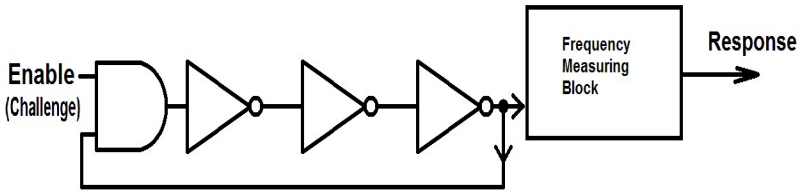
PUF Classification: Weak vs Strong

Strong PUF

- May have many possible challenge response pairs (CRPs), e.g. Arbiter PUF.
- With access to the CRPs, it should be infeasible to model the system and determine the CRPs of a strong PUF
- Applications include:
  - Challenge-response authentication
  - Anti-cloning

15

How to Design a Physically Unclonable Function?



16

Ring Oscillator is one of the most used circuits for CS. If I enable RO, it will start to oscillate with a certain frequency I can measure (enable is chl). Very simple circuit of course. Why can I consider oscillation frequency as a PUF?

$$f_{osc} = \frac{1}{N(\tau_{HL}+\tau_{LH})} = \frac{1}{2N\tau_p} \quad \text{where } \tau_p = \frac{\tau_{HL}+\tau_{LH}}{2}$$

Propagation delay depends on parasitic loads, current flowing through transistors. Current depends on threshold voltage, mobility, Cox (dielectric oxide permittivity) $C_{ox} = \frac{\varepsilon_{ox}}{t_{ox}}$. So everything will be different, and if I'm precise enough w/ measurements great. ↳ width

JMP 17

Hey Giovanni! The **Cox** parameter in MOSFETs refers to the **oxide capacitance per unit area**, and it's a key part of how the MOSFET operates, especially in the context of gate control over the channel.

Cox is given by:

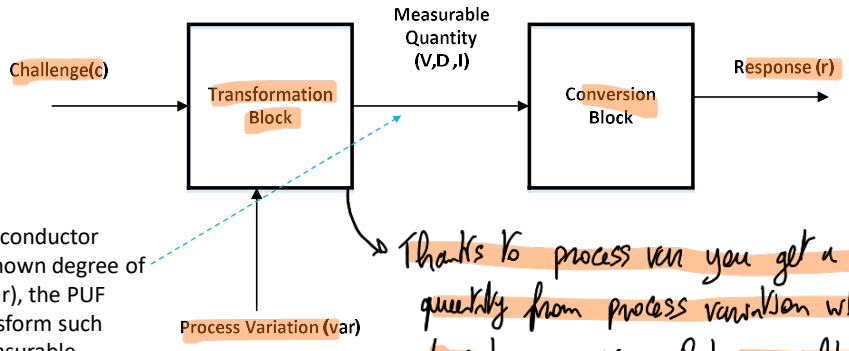
Cox = εox / tox

Where:

- **εox** is the permittivity of the gate oxide (usually silicon dioxide, SiO_2)
- **tox** is the thickness of the oxide layer

A Generic Architecture for Silicon-based PUFs

You have 2 funct. blocks that give you behaviour of PUF

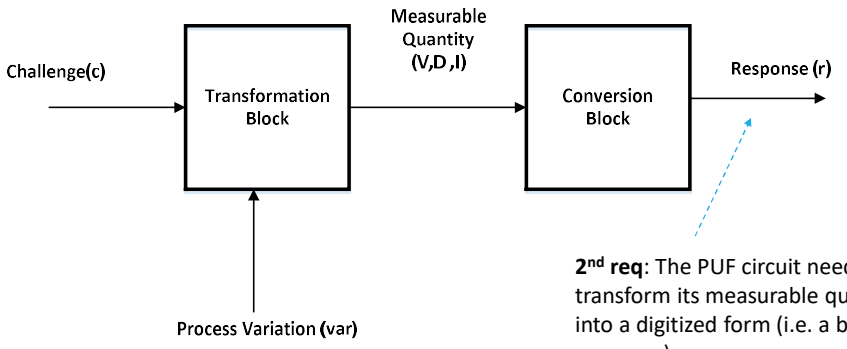


1st req: Given a semiconductor technology with a known degree of process variation (var), the PUF circuits need to transform such variations into a measurable quantity (mq)

Thanks to process var you get a measurable quantity from process variation whose value depends on pVar. But we would like to characterize devices with a digital sequence: so a conversion block gives a unique digital response (sequence of 0s and 1s)

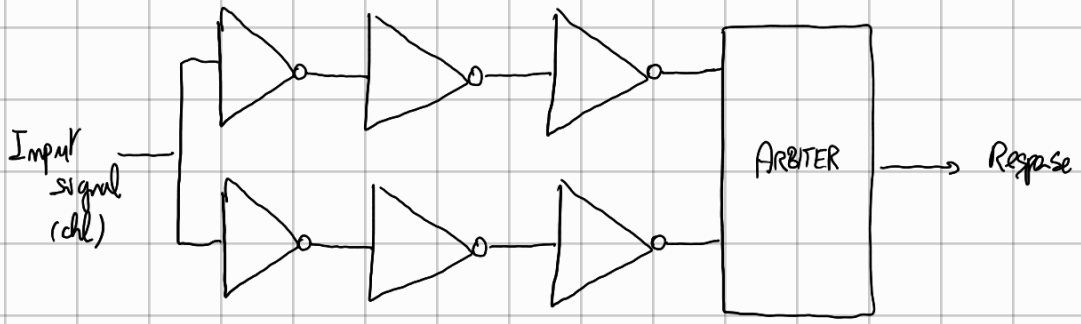
17

A Generic Architecture for Silicon-based PUFs



2nd req: The PUF circuit needs to transform its measurable quantity (mq) into a digitized form (i.e. a binary response).

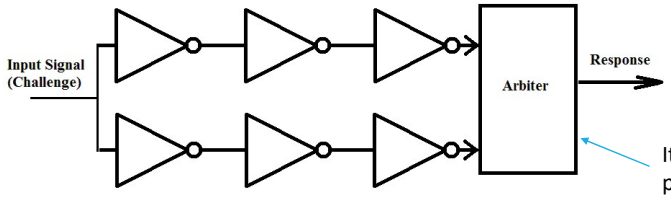
18



We have an Arbiter PUF: a series (cascade) of inverters (not RO) and an arbiter block. As a challenge we apply an impulse (a step). We go from a stable condition, and I get after the impulse the propagation of input signal. Inverters will behave differently because of process. Transition will arrive first either up or down: arbiter has to decide which one arrived first and give a 0 or 1 response accordingly. So two paths that race, an arbiter that decides.
Btw I also expect within the same cascade inverters that behave differently.
If I implement something a bit more complex, a circuit with two paths that race, and if I'm able to differentiate those two paths, I can expect that arbiter result can identify a device. Here chl is invented but we can have something that works.
Of course, same challenges to same device even on different conditions must give same response.
JMP 20

Delay-Based Designs

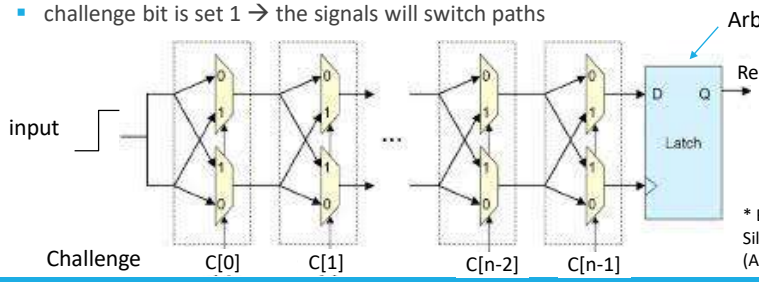
- An **Arbiter PUF** consists of **two digital paths** with **identical nominal delay** and an **arbiter**.
- When an input signal is applied, it will propagate through these two paths, and **arrive to the inputs of the arbiter at slightly different moments** because the **intra-die variability**, the arbiter outputs a logic "1" if signal on path 1 wins the race and a logic "0" otherwise: **1-bit response**
- The **output of this design will be different when implemented on different chip**, due to in**ter-die process variations**



19

Arbiter PUF*

- The two (nominally identical) paths are simultaneously asserted with an input pulse.
- Each path consists of a set of stages with each stage containing a switch circuit, which is composed of two MUXs that are controlled by a challenge bit
- The challenge bit determines which paths the input signals take within each switch:
  - challenge bit set as 0 → the input signals will continue to the output along their current paths;
  - challenge bit is set 1 → the signals will switch paths



CRPs = 2^n → strong PUF

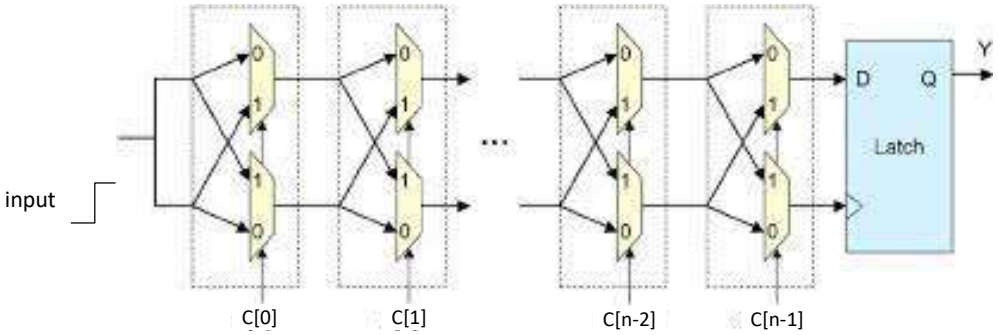
* B. Gassend, D. Clarke, M. Van Dijk, S. Devadas, Silicon physical random functions, in *Proc. CCS* (ACM, 2002), pp. 148–160

20

Propagation paths are done with MUXs up and down. And control bit is given by challenge. Out of a mux contributes differently for each MUX. Out up goes to 0 to next mux and 1 to down mux. If C[i]=0, signal propagates horizontally. If C[i]=1, then we cross paths for muxes. So I can activate different prop. paths that can get activated so different behaviour.

So signal will propagate to arbiter. How to implement arbiter? With a D latch. Why? Input that comes from upper mux chain is D, while down chain is clock. If we start with stable 0 condition what happens? I stimulate input with 0→1 transition. If downwards path comes first, a 0 is sampled. If upwards comes first, a 1 is sampled.
If I need K bits of response, given n muxes in cascade, configurations are 2^n (Strong PUF). So I apply K challenges to get my response.
But we have a problem: those two paths have very little differences. Those little differences could violate setup and hold time of D FF. which means we can work in metastable situations, so with same challenge I can give different responses. This can make PUF unusable for inconsistencies.
Another limitation: I want to differentiate prop. delay only from PVT, not because of interconnections (okay, doable in ASIC, but not in FPGA).
In an FPGA propagation delay can depend more on routing than on PVT. So this won't work well there.
On top of this, it was shown that this kind of arbiter PUF is weak: through Machine Learning, after observing a high number of responses I can start to predict responses of unseen challenges (modelling attacks).
JMP 24

Arbiter PUF*



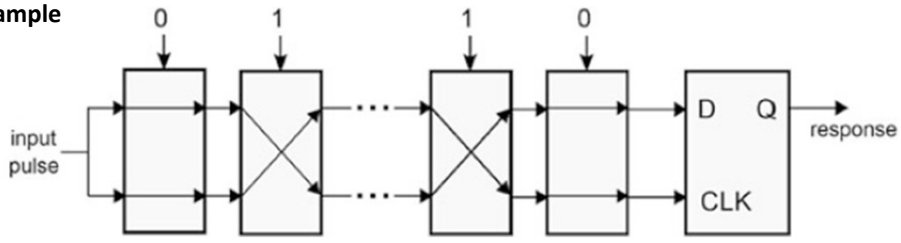
* B. Gassend, D. Clarke, M. Van Dijk, S. Devadas, Silicon physical random functions, in *Proc. CCS* (ACM, 2002), pp. 148–160

21

Arbiter PUF

- **If the pulse reaches the output of the first path faster**, the **Arbiter outputs a logic 1**. **Otherwise**, it outputs a **logic 0**
- The output/response depends on the delay present in both paths and is a function of the variations experienced by an IC during fabrication

Example



22

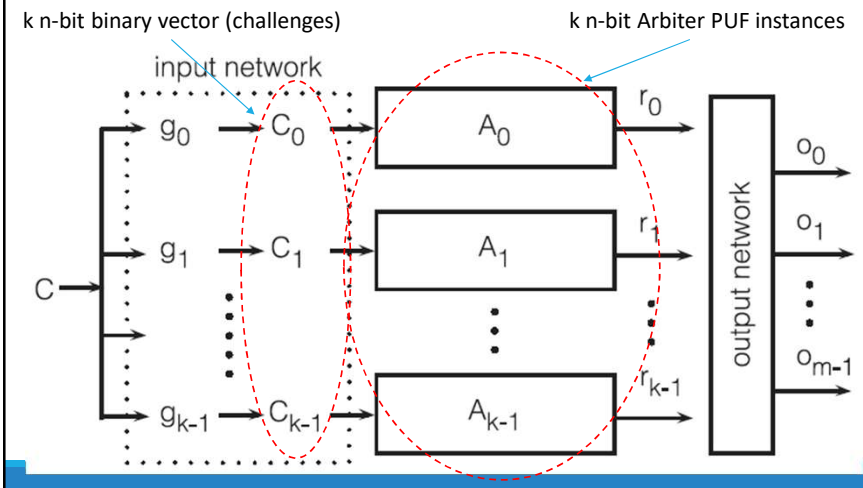
Arbiter PUF: Limitations

- A robust Arbiter PUF is tough to achieve in practice:
  - First, to generate a correct response, the **timing difference** between the two paths has to satisfy the **setup time and hold time requirements of the D-latch**
  - Second, the **routing** of both paths must be **perfectly symmetric** which can be difficult to obtain in practice, especially in FPGAs
- **Without symmetric routing**, **the PUF response bits are biased** towards one value (0 or 1)
- Finally, it has been shown that after observing a number of CRPs, simple **machine-learning techniques** can be used to predict PUF responses to unseen challenges with relatively high accuracy → this flaw could **allow attackers to determine a PUF response** to a new challenge without being in possession of the IC

23

How to do something a bit more secure? We have a series of arbiter PUFs (suppose we solve metastability), k of them, and an input network and output one. To input network I apply a combination of challenges (non observable) that are going to be applied to arbiter PUFs which will give results not observable by external world, and output network gives a series of output.

Secure Arbiter PUF

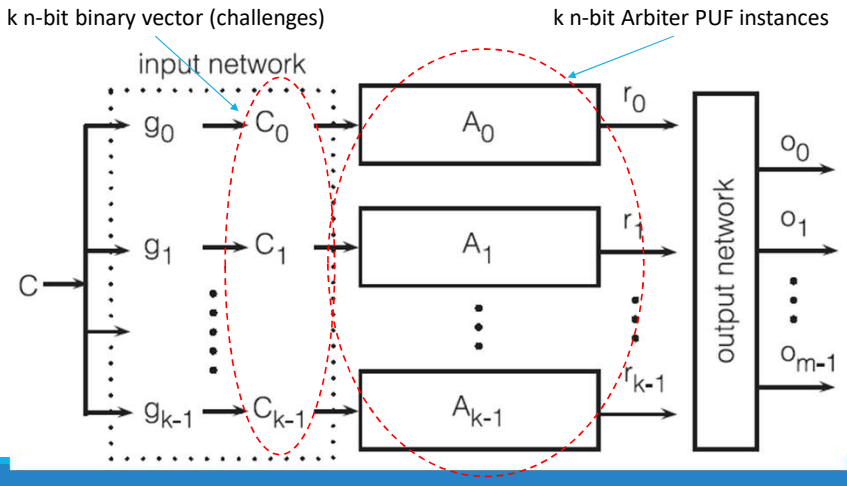


- This secure PUF uses many instances of the "insecure" Arbiter PUF that could be modelled by machine learning attacks Secure PUF
- Input network produces a set of k challenges, each composed by n bits

24

* g_i represent function that produce C_is.

Secure Arbiter PUF



- Output network: m-bit outputs $(o_1 \ldots, o_{m-1})$ from k-bit output $(r_1, \ldots, r_{k-1})$ generated by the k APUFs

parameter for confusion

$o_i = \oplus_{j=0}^{x-1} r_{(i+s+j) \bmod k}$

index of responses to combine xor

for $i = 0 \ldots m-1$

$r_i = A_i(C_i) \quad i = 0 \ldots k-1$

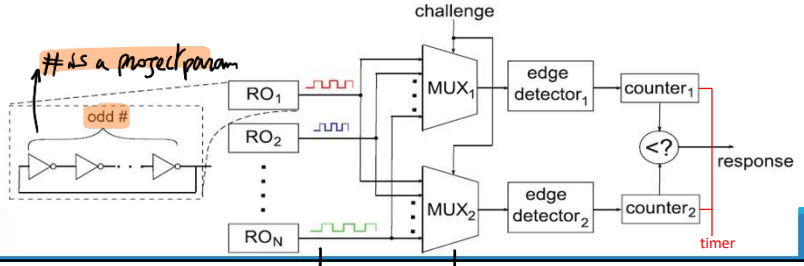
$x < k$ and s are security parameters chosen by the designer

25

So I observe linear combination of arbiter output. This makes modelling attacks harder.

Ring Oscillator PUF

- A generic **ring oscillator based PUF** is made of two multiplexers, two counters, one comparator and *K* ring oscillators
- Each ring oscillates at a **unique frequency** depending on the characteristics of each of its inverters, the two **multiplexers select two ROs to compare**. An **edge detecto**r detects the **rising edges** in output oscillations and the two **counters count the number of edges** in a **fixed time interval**
- At the **end of the interval**, the outputs of the two **counters are compared**, and depending on which of the two counters has the highest value, the **output** of the PUF is **set to 0 or 1**



26

We have n of them, all ideally identical

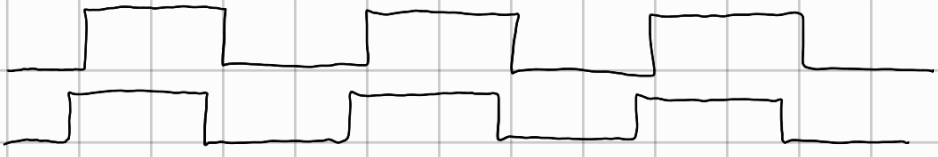
We connect them to pairs of MUXs, all of them, connected to same challenge. ①

NOTE: what can you do about metastability? You can select a subset of challenges by which the delays are sufficiently different to avoid metastability. You still want to maintain strength of PUF though.

Other solution is RO PUF.

(1) Since chal is same, I don't want to select same RO with two MUXes, so I need to make sure to not do that.

All ROs will have different frequencies, and a pair of ROs will be propagated. I need to compare this pair w/ something that gives info on frequency difference. I can use an edge detector connected to a counter. Given a sufficiently long time interval, I measure different edges and compare them! You will have something like:

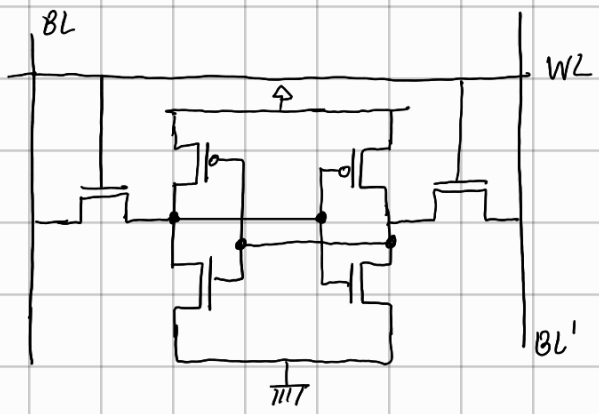


On a long enough time interval I can have a difference. So the result gives me a bit of response. With multiple challenges I can get multiple bits.

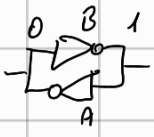
Typically we expect tens of inverters in ROs and tens of ROs.

What can we say about SRAM PUFs?

SRAM cell:



We use WL to activate pass transistors to connect to our cell. And our bitline will read value stored in the SRAM cell (while BL' the inverted value). We have a stable condition in which bistable has 0 and 1:

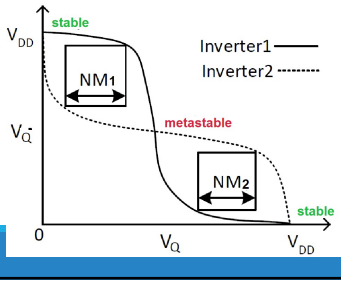


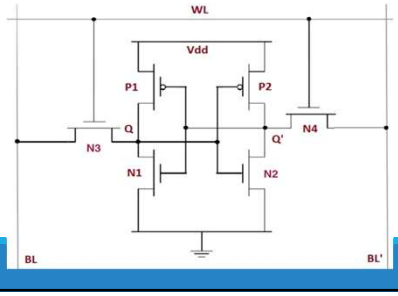
Stability.

My working circuit will force this stability. But suppose my bistable has no power supply and then I turn on device (so I give V_{DD} to the correct value). What happens? Eventually, after metastability my cell will evolve on a stable state (either our inverter A = 1 or 0). It depends on process and conductivity of cells, that will eventually solve electrical conflict.
So memory cell will have BL = 0 and BL' = 1 or vice versa. If we do this with an entire memory word I have a signature: If I repeat this process I expect same behaviour. And here I have my weak PUF, because I don't have an actual challenge (maybe just memory word address. Weak PUF because chl-rsp pairs increases linearly with system complexity (I add another memory line).

SRAM PUFs

- A typical SRAM cell is composed of **two cross-coupled inverters** (P1, N1, P2, N2) and two N-type **access transistors** (N3, N4).
- The two inverters have two stable states logic '1' or logic '0',
- Each inverter drives one of the two state nodes, Q or Q'
- When this **cell** is **powered up**, the two cross-coupled inverters enter a "**power struggle**"; the winner will be ultimately decided by the **difference in the driving strength** of the MOSFETs in the **cross-coupled inverters**

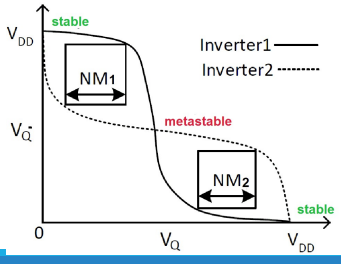


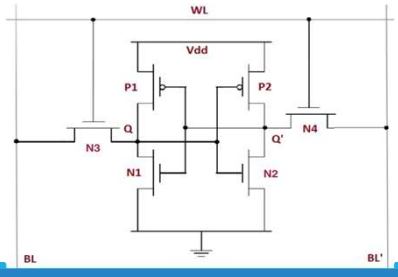


27

SRAM PUFs

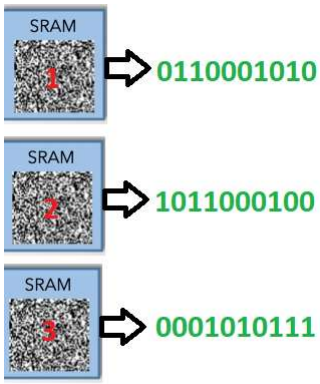
- Although those **transistors are designed to have identical nominal sizes**, random intra-die variations in the **silicon manufacturing process** ensure that **one inverter** has a **stronger** driving current than that of the other inverter, this helps define the unique initial start-up value for the cell





28

SRAM PUFs



- The power up-value of a whole SRAM block is also unique for each device due to inter-die variability
- SRAM PUF exploits this to generates a unique response from each device

29

Any questions so far?

30

Brief outline

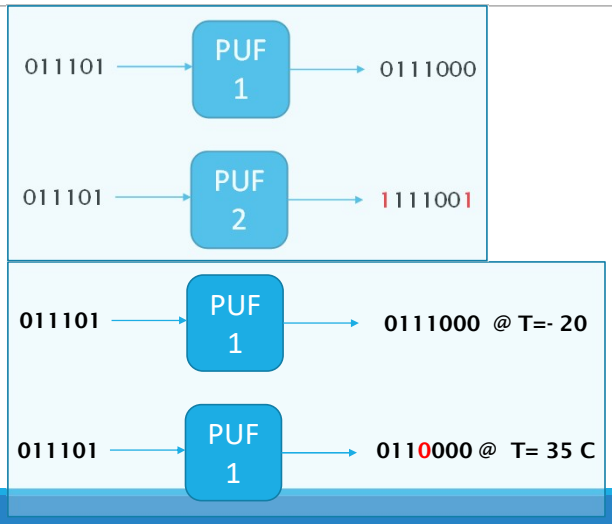
- Design and evaluation principles of PUF
- **Quality metrics for PUF evaluation**
- PUFs generating/storing cryptographic keys
- PUFs for entity authentication protocols
- PUFs for hardware metering

Examples of possible applications

31

Quality Metrics

- **Uniqueness** is a measure of the ability of a device to generate **unique IDs**
- **Reliability** is of a measure of the ability of the PUF to generate a **consistent response** for the same challenge, given the **environmental variability** (and **aging**)



32

- We want PUFs to be unclonable, to be reliable (RELIABILITY) (same chl → same response on same chip) and unique (UNIQUENESS): ability for a device to have unique signature: same chl on different devices must give different responses.

NOTE: Reliability has to be ensured in different working conditions and aging conditions (results should be the same).

Hamming distance: # of bits in two words that need to be changed to get from one bit pattern to another.

Rigorously: $d(a,b)$ between two words $a=(a_i)$ and $b=(b_i)$ of length n is the number of positions where they differ: the number of i such that $a_i \neq b_i$.

Hamming Distance

The Hamming distance is the number of bits that have to be changed to get from one bit pattern to another

Example: 1001**01**01 & 1001**10**01 have a Hamming distance of 2

A more rigorous definition: the Hamming distance *d*(a, b) between two words **a = (a_i)** and **b = (b_i)** of length **n** is defined to be the number of positions where they differ, that is, the number of *i* such that **$a_i \neq b_i$.**

Quiz: Find the Hamming distance between:

- **a= 00** and **b= 11**
- **a=0100 and b= 0110**

33

Quality Metrics: Uniqueness

Inter chip Hamming distance is a parameter we will use

- **Uniqueness** is a measure of the ability of a device to generate **unique IDs**

  In other words: it is the measure of the ability of one **PUF instance** to be **uniquely distinguished from other PUF with the same structure implemented on different chips**
  → Ideally, the value of uniqueness is 50%

- The Hamming Distance (HD) is used to evaluate the uniqueness performance and is called the '**Inter-chip HD**'. If two chips, *i* and *j* ($i \neq j$), have ***n*-bit responses**, $R_i(n)$ and $R_j(n)$, respectively, for the challenge *C*, the **average inter-chip *HD*** among ***k* chips** is defined as:

$$HD_{INTER} = \frac{2}{k(k-1)} \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} \frac{HD\left(R_i(n), R_j(n)\right)}{n} \times 100\%$$

Average on all possible combinations of chips to check

HD of two chips with same challenge response length

34

Given uniqueness def, I expect to have all different responses after a chl application. Ideally, taking Hamming distance to measure uniqueness, ideal value of different bits I get is 50% in my global set: I take two answers and expect the HD on average to be 50%.

I measure uniqueness with inter chip HD: I measure average HD I get.

I apply same challenge to k PUFs I'm considering, check all possible HD and take average. In practice I expect something close to 50%. 45-55% is acceptable.

Quality Metrics: Reliability

- **Reliability** is of a measure of the ability of the PUF to generate a **consistent response *R*** for a challenge *C*., given changes in ambient temperatures and/or voltage supply fluctuations
- The Hamming Distance (HD) is used to evaluate the reliability performance and is called the '**Intra-chip HD**'. If a single chip, represented as *i*, has ***n*-bit reference response** *Ri*(*n*) from the chip *i* at normal operating conditions (at room temperature using the normal supply voltage) and the same *n*-bit response obtained at different conditions *R'i* (*n*) respectively for the challenge *C*, the **average intra-chip HD** for ***k* samples/chips** is defined as:

$$HD_{INTRA} = \frac{1}{k}\sum_{i=1}^{k}\frac{HD\left(R_i(n), R_i'(n)\right)}{n} \times 100\%$$

Ideally HD should be 0.

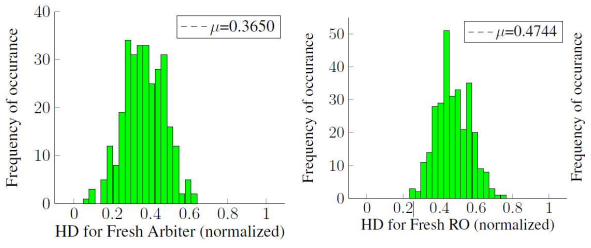
From the intra-chip HD value, the reliability of a PUF can be defined as:

$$reliability = 100\% - HD_{INTRA}$$

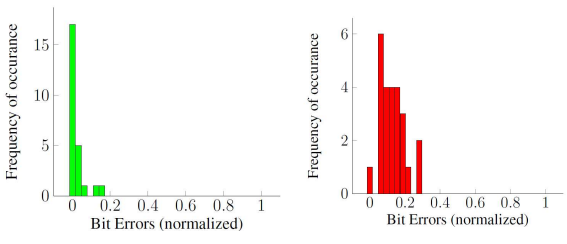
35

Intra-chip HD: I work with one single chip with different samples.
This gives me a probability of measure being reliable.
NOTE: If I don't do any post processing on data gathered from PUFs we have seen we will be distant from ideal values (usually we use error correction)

Quality Metrics: Uniqueness



Uniqueness Metric of PUF



Ageing-Induced Bit errors after 10 years of operation

a) Arbiter; b) Ring Oscillator

36

Thank you!

Any questions?

37