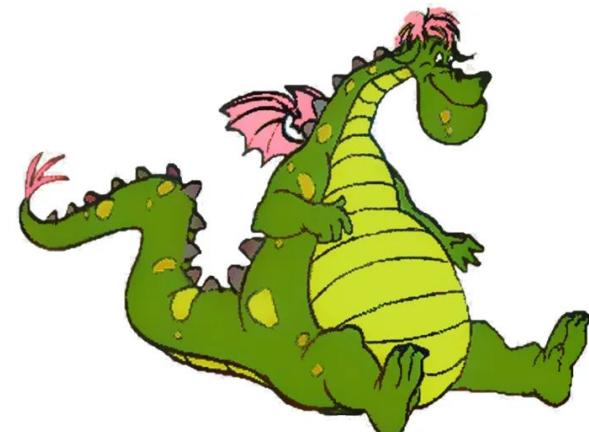
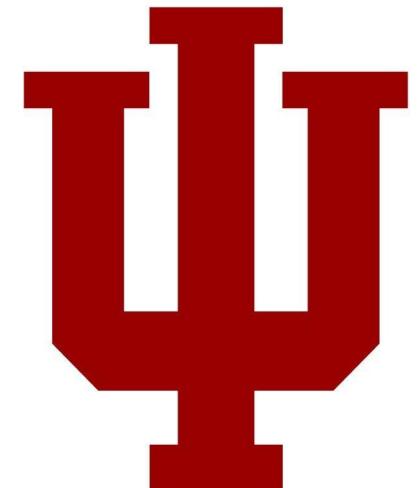


05 Physically Uncloneable Functions (PUFs)

Engr 399/599: Hardware Security
Grant Skipper, PhD.
Indiana University



^ not this



Adapted from: Mark Tehranipoor of University of Florida

Course Website

enr599.github.io

Write that down!

Agenda

- Housekeeping!
- Warmup!
- Review VLSI
- PUFs!
- Maybe TRNG?
- Project #1 Due at Midnight!

Side Quest

<https://www.wired.com/story/snoo-smart-bassinet-vulnerabilities-shaking-loud-noise/>

Old Story - 2020-2022

What is a critical system?

Security is often key to product safety. Sometimes difficult to really know the impact unless you are an integrator.



how many g's can baby withstand

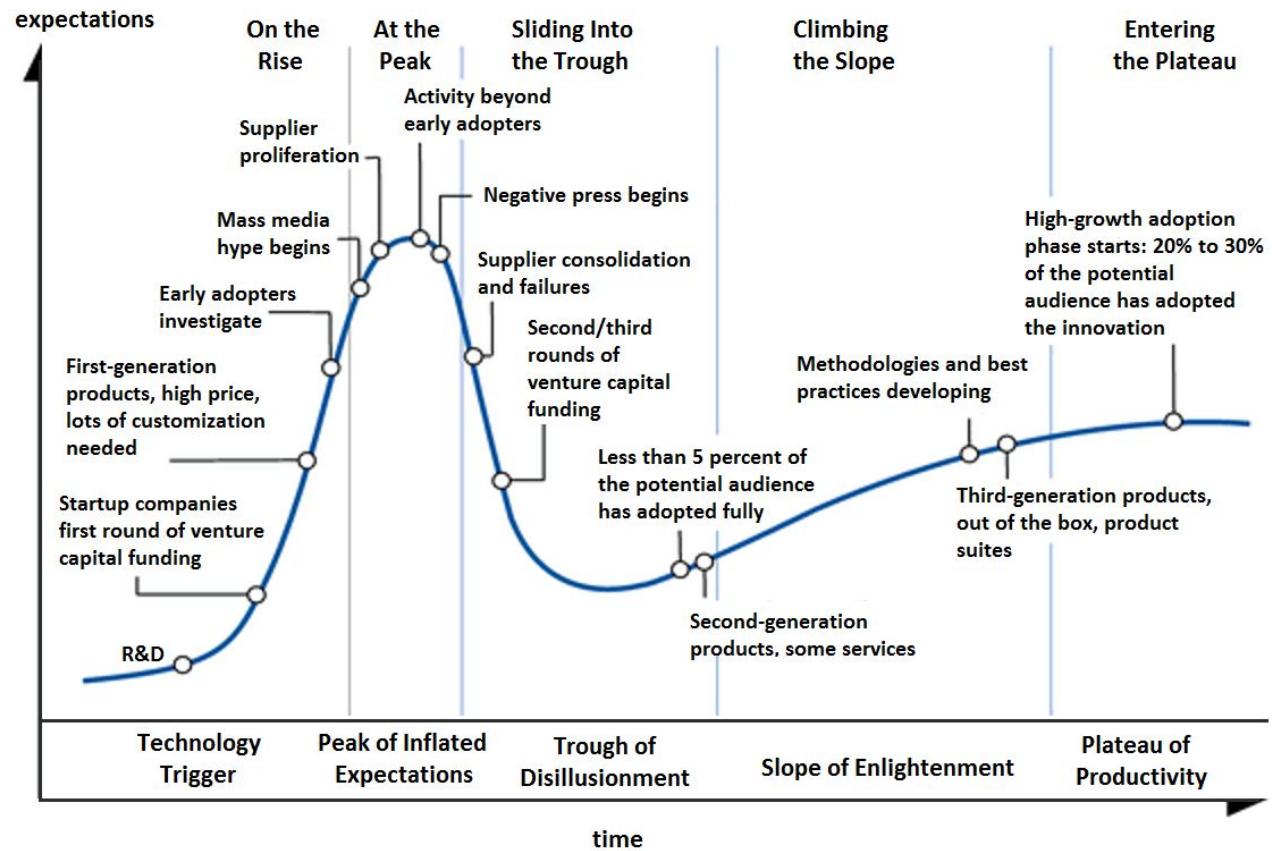
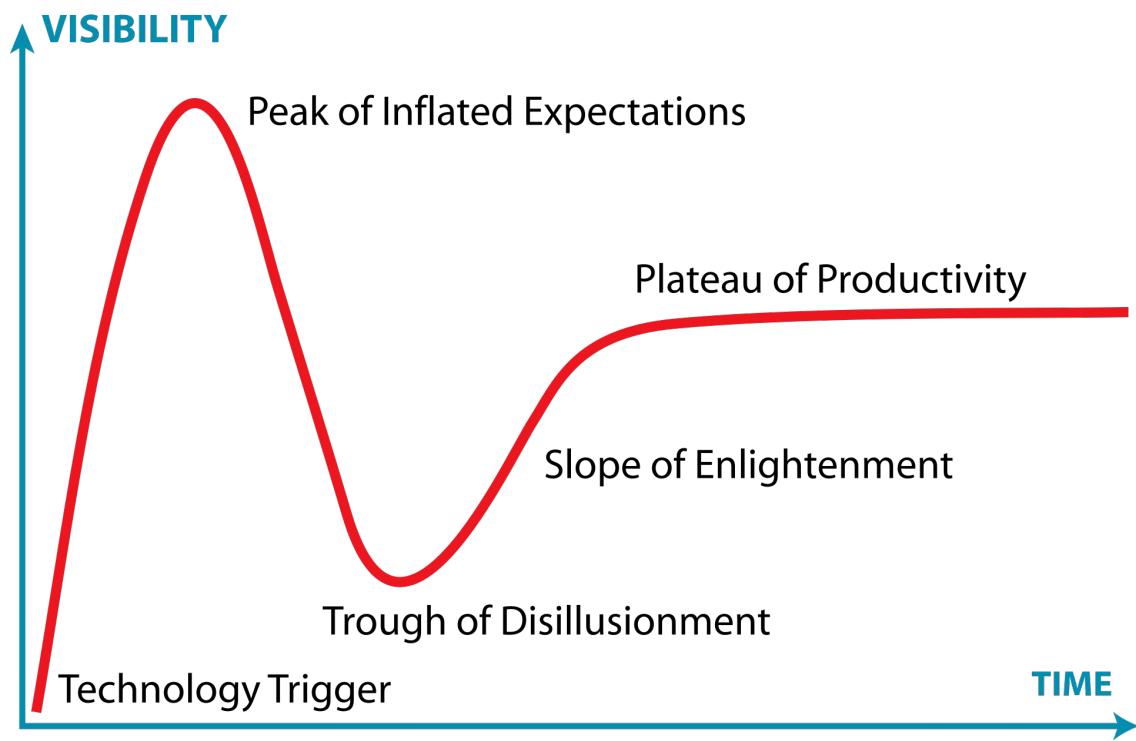
AI Mode All Images Videos Shopping Short videos News More Tools

AI Overview

WAT?

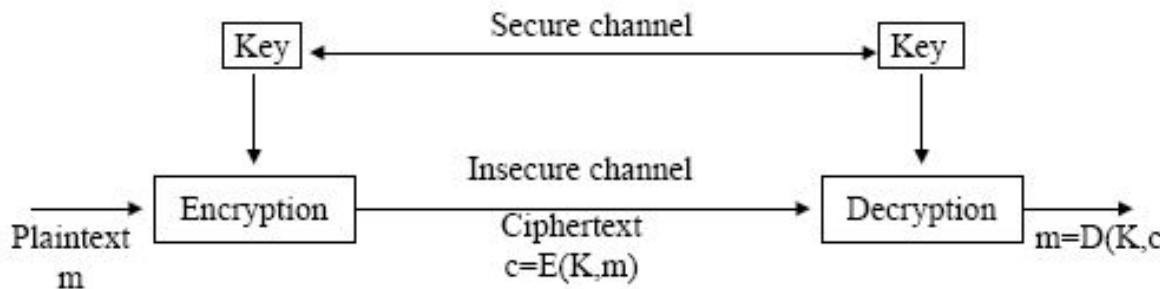
Infants are highly vulnerable to G-force, with violent shaking causing head accelerations around 10–18.6 G-forces, which can cause severe brain injury, bleeding, and death. While a 50th percentile child might withstand 50 g's in controlled, short-duration impact tests (e.g., car seats), any sudden, high-magnitude force or sustained G-force is extremely dangerous. MDPI +3

Gartner Hype Cycle



Symmetric and Asymmetric Cryptosystems (1)

- Symmetric encryption = **secret key encryption**
 - $K_E = K_D$ — called a **secret key** or a **private key**
 - Only sender S and receiver R know the key



[cf. J. Leiwo]

- As long as the key remains secret, it also provides **authentication** (= proof of sender's identity)

General Security



© 2002 by Paul Kocher

Digital Storage



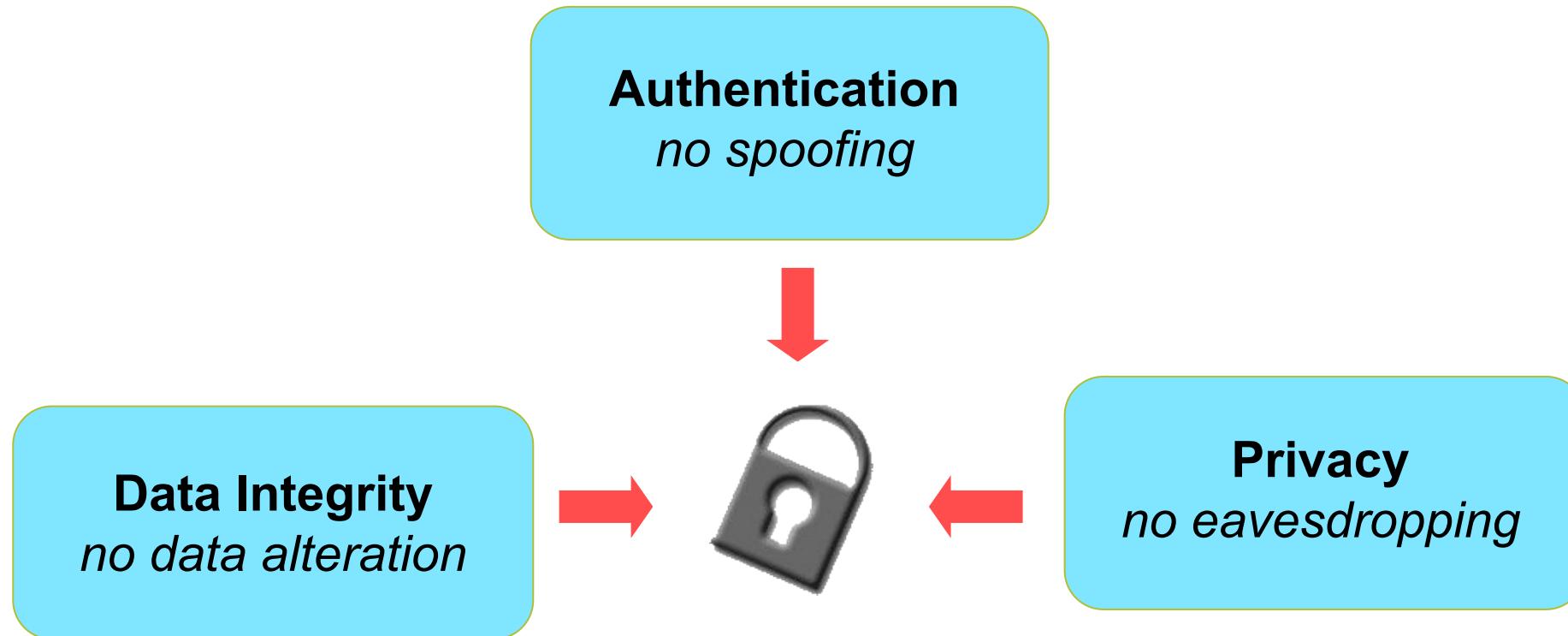
In traditional methods, secret keys are stored digitally in a nonvolatile memory which is always vulnerable based on hardware implementation and key storage.

Authentication



For extremely resource constrained platforms such as RFIDs, even simple cryptographic operations can be too costly.

What We Want to Achieve?



Attacks



Software-only protection is not enough. Non-volatile memory technologies are vulnerable to invasive attack as secrets always exist in digital form

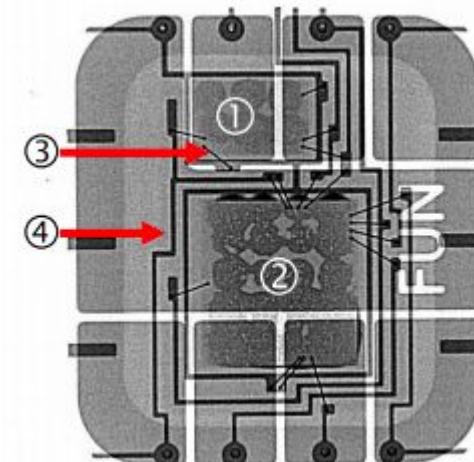
Threat Model

■ Attacker goals

- To get the crypto keys stored in RAM or ROM
- To learn the secret crypto algorithm used
- To obtain other information stored into the chip (e.g. PINs)
- To modify information on the card (e.g. calling card balance)

Over \$680,000 stolen via a clever man-in-the-middle attack on chip cards in 2011.

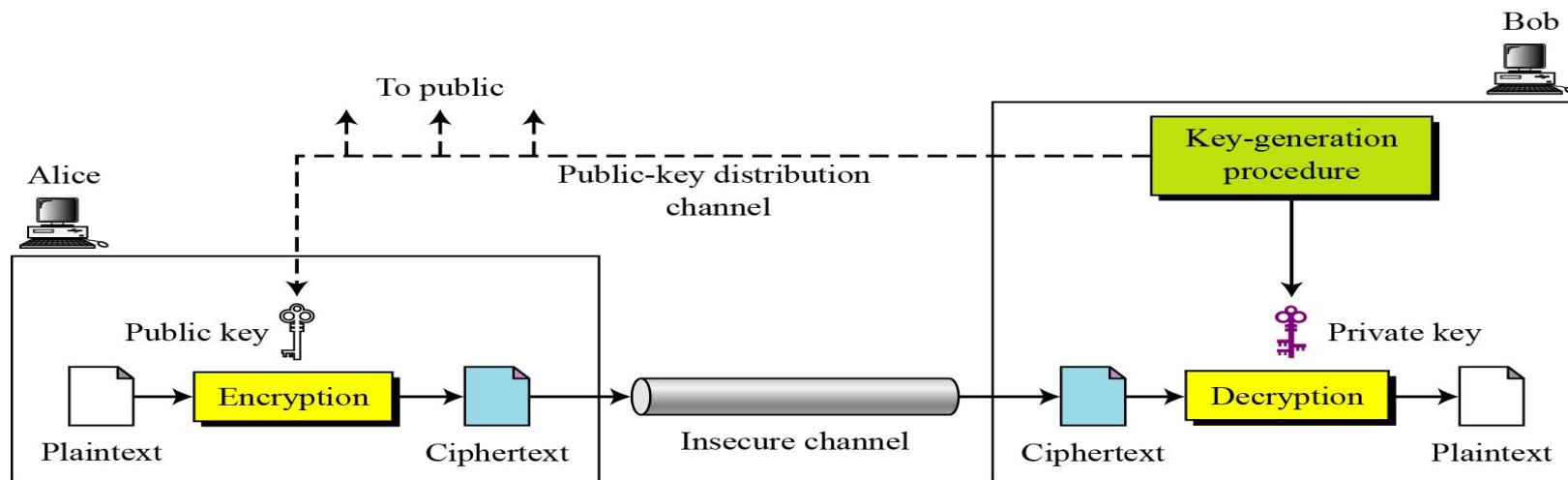
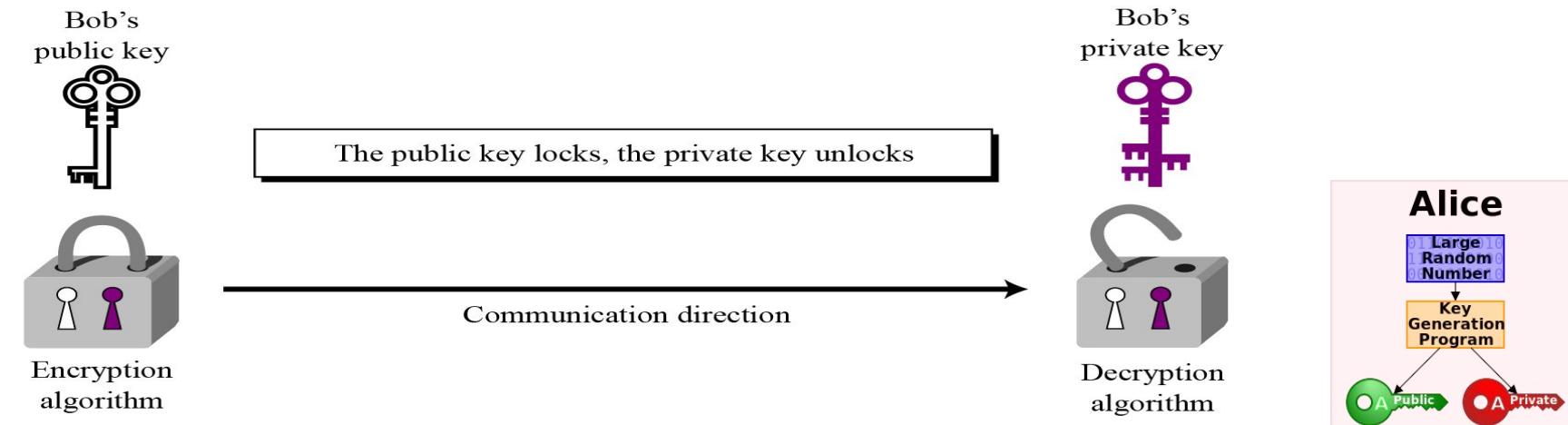
<https://arstechnica.com/tech-policy/2015/10/how-a-criminal-ring-defeated-the-secure-chip-and-pin-credit-cards/>



Basic Terminologies

- Keys are rules used in algorithms to convert a document into a secret document
- Keys are of two types:
 - Symmetric
 - Asymmetric
- A key is symmetric if the same key is used both for encryption and decryption
- A key is asymmetric if different keys are used for encryption and decryption

Asymmetric Security



Security

- Asymmetry b/w the information (secret)
- **One-way functions (Trap Doors)**
 - Easy to evaluate in one direction but hard to reverse in the other
 - E.g., multiplying large prime number as opposed to factoring them
- **One-way hash functions**
 - Maps a variable length input to a fixed length output
 - **Avalanche property:** changing one bit in the input alters nearly half of the output bits
 - Pre-image resistant, collision resistant
 - Usage: digital signature, secured password storage, file identification, and message authentication code

Challenges of algorithmic (mathematical) one-way functions

- **Technological**

- Massive number of parallel devices broke DES
 - Reverse-engineering of secure processors

- **Fundamental**

- There is no proof that attacks do not exist
 - E.g., quantum computers could factor two large prime numbers in polynomial time

- **Practical**

- Embedded systems applications

Can be modeled? ...

Solution -- POWF

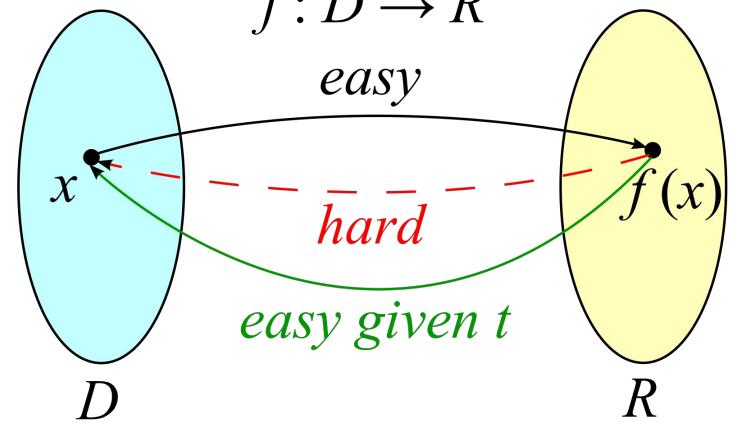
- Use the **chaotic physical structures** that are hard to model instead of mathematical one-way functions!
- Physical One Way Functions (POWF)
 - inexpensive to fabricate
 - Prohibitively difficult to duplicate
 - No compact mathematical representation
 - Intrinsically tamper-resistant

Trapdoor Function!

$$(f, t) = \text{Gen}(1^n)$$

$$f: D \rightarrow R$$

easy



IBM 4758

Problem:

Storing digital information in a device in a way that is resistant to physical attack is difficult and expensive.



IBM 4758

Tamper-proof package containing a secure processor which has a secret key and memory

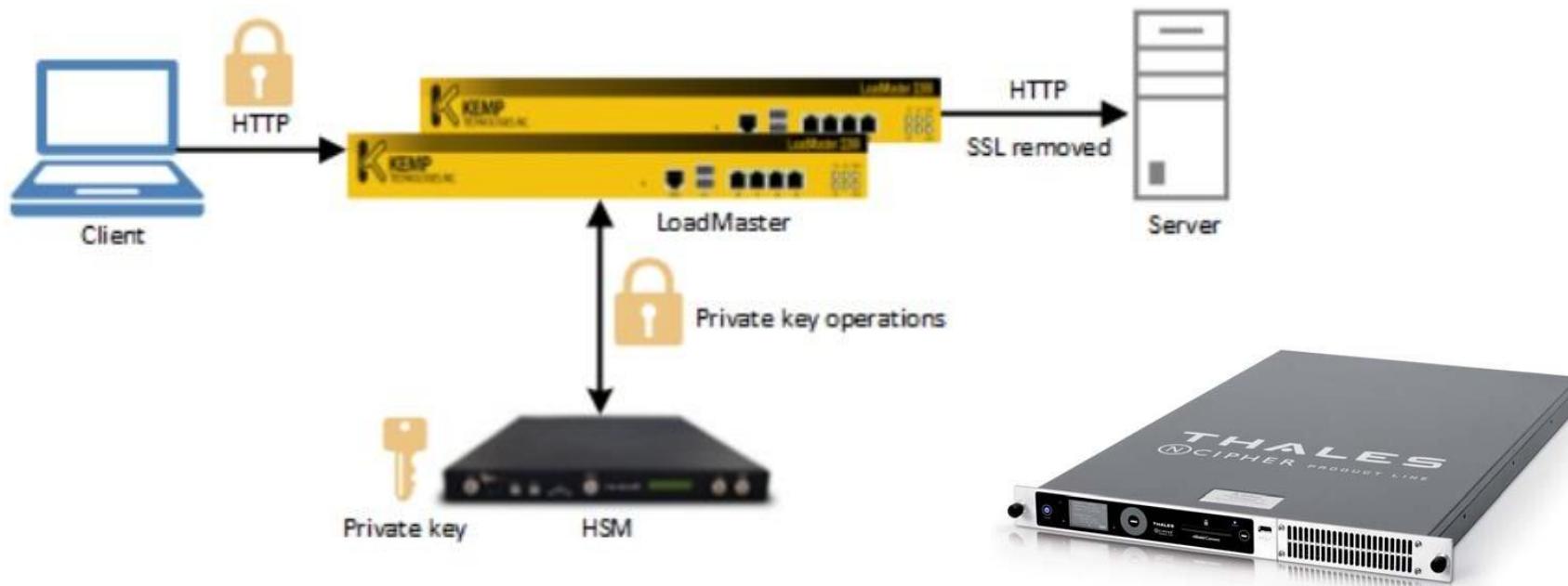
Tens of sensors, resistance, temperature, voltage, etc.

Continually battery-powered

~ \$3000 for a 99 MHz processor and 128MB of memory

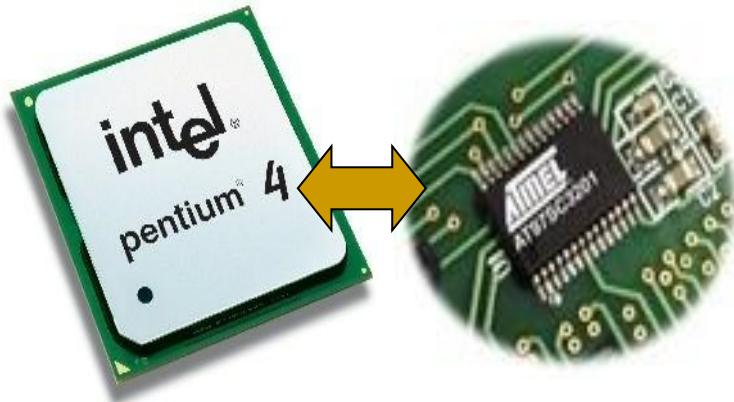
HSM

A **hardware security module (HSM)** is a physical computing device that safeguards and manages digital keys for strong authentication and provides crypto processing. These modules traditionally come in the form of a plug-in card or an external device that attaches directly to a computer or network server. - Wikipedia



TPM

A **Trusted Platform Module (TPM)** is a specialized chip on an endpoint device that stores RSA encryption keys specific to the host system for hardware authentication. Each TPM chip contains an RSA key pair called the Endorsement Key (EK). -- Wikipedia



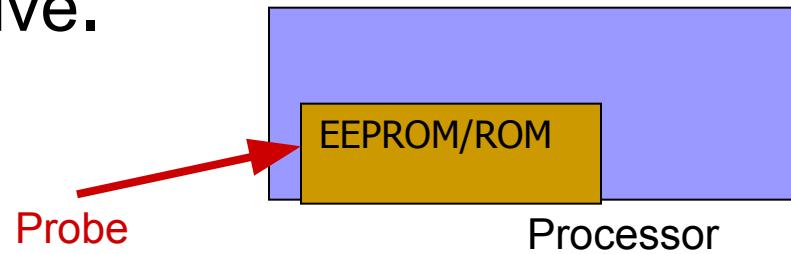
A separate chip (TPM) for security functions

Decrypted “secondary” keys can be read out from the bus

[Uprooting Trust Paper](#)

Problem

Storing **digital** information in a device in a way that is resistant to **physical attacks** is difficult and expensive.



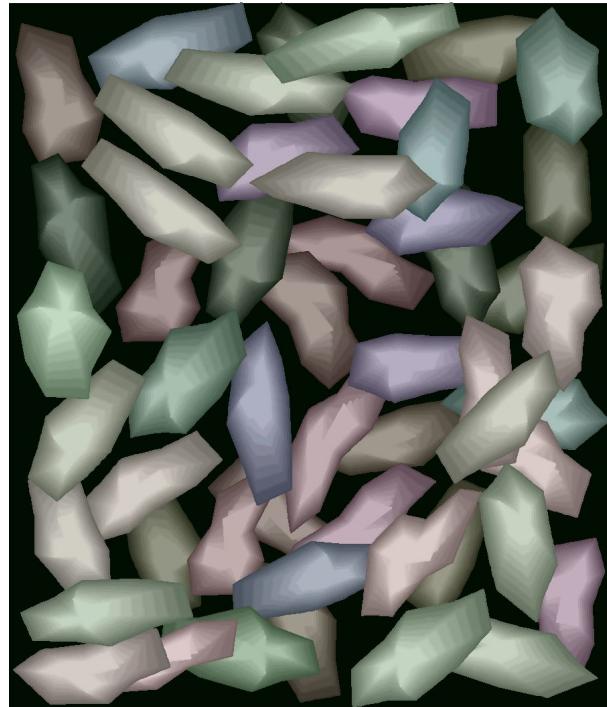
- Adversaries can physically extract **secret keys** from EEPROM while processor is off
- Trusted party must embed and test secret keys in a secure location
- EEPROM adds additional complexity to manufacturing

Feature: Process Variation

- Do we expect process variation (length, widths, oxide thickness) in circuit and system?
 - Impact circuit performance
 - Functional failure
 - Major obstacle to the continued scaling of integrated-circuit technology in the sub-45 nm regime
- Process variations can be turned into a feature rather than a problem?
 - Each IC has unique properties

Solution

Extract key information from a complex physical system.

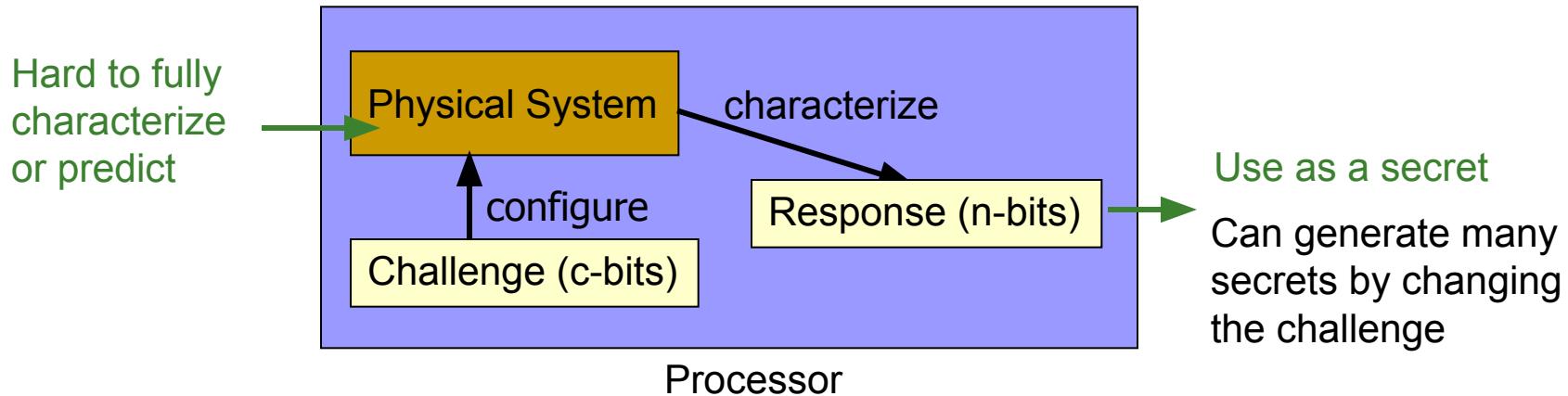


“In information theory, the entropy of a random variable quantifies the average level of uncertainty or information associated with the variable's potential states or possible outcomes.”
- Wikipedia

Devadas, et. al, DAC02

Physical Unclonable/Random Functions (PUFs)

- Generate keys from a complex physical system



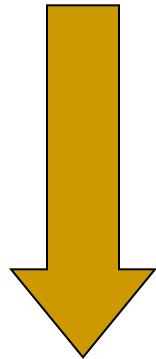
- Security Advantage
 - Keys are generated on demand □ No non-volatile secrets
 - No need to program the secret
 - Can generate multiple master keys
- What can be hard to predict, but easy to measure?

Definition

A Physical Random Function or **Physical Unclonable Function (PUF)** is a function that is:

- Based on a physical system
- Easy to evaluate (using the physical system)
- Its output looks like a random function
- Unpredictable even for an attacker with physical access

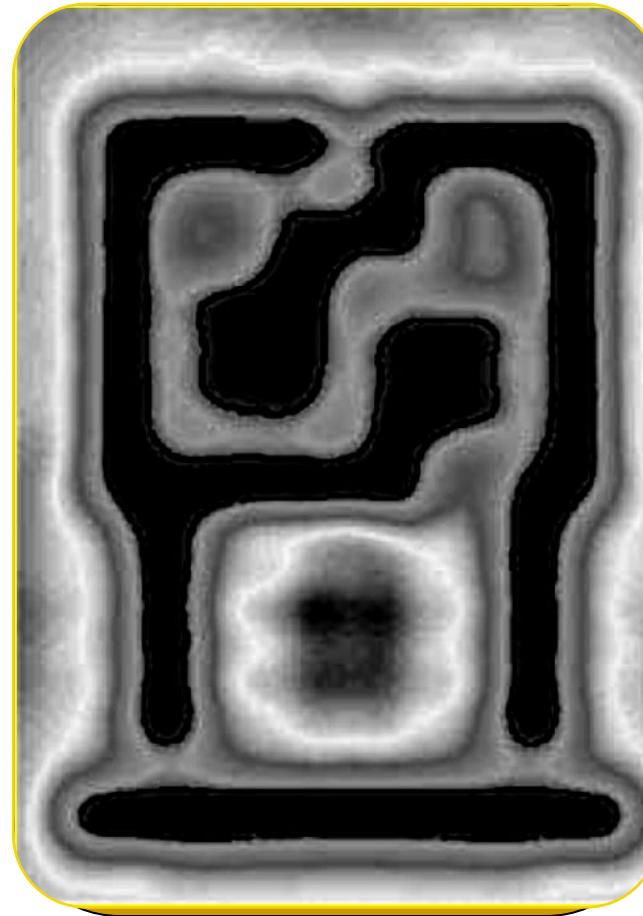
Sub-Wavelength WYSINWYG



What You See Is Not What You Get

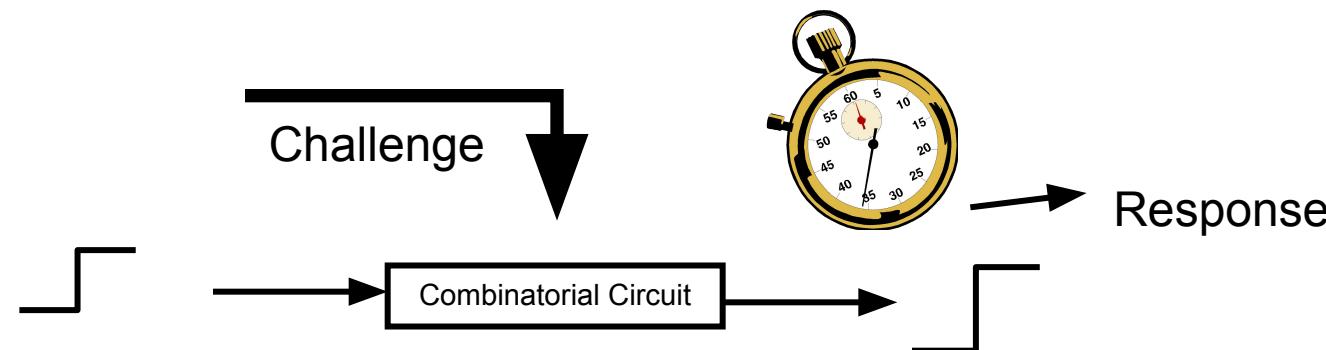
Process variations

No two transistors have the same parameters

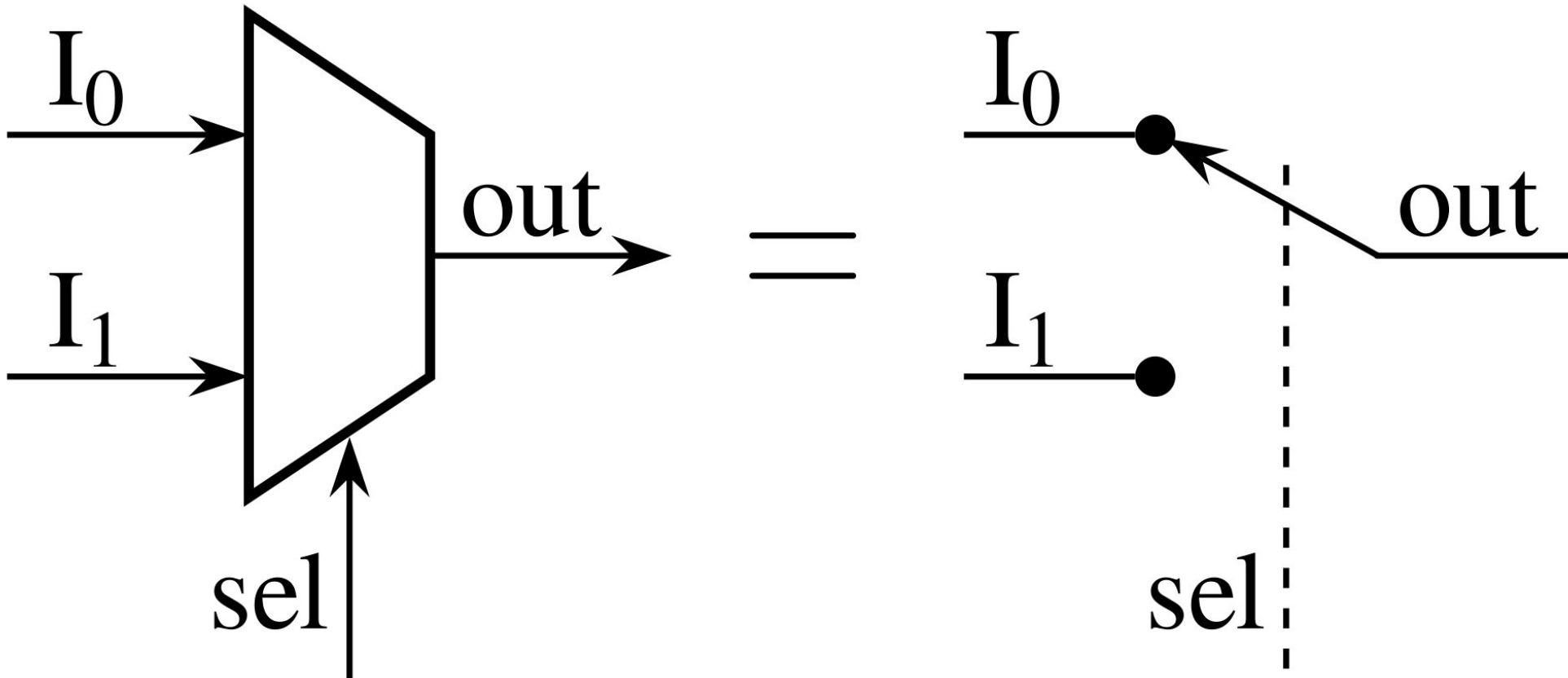


Silicon PUF – Proof of Concept

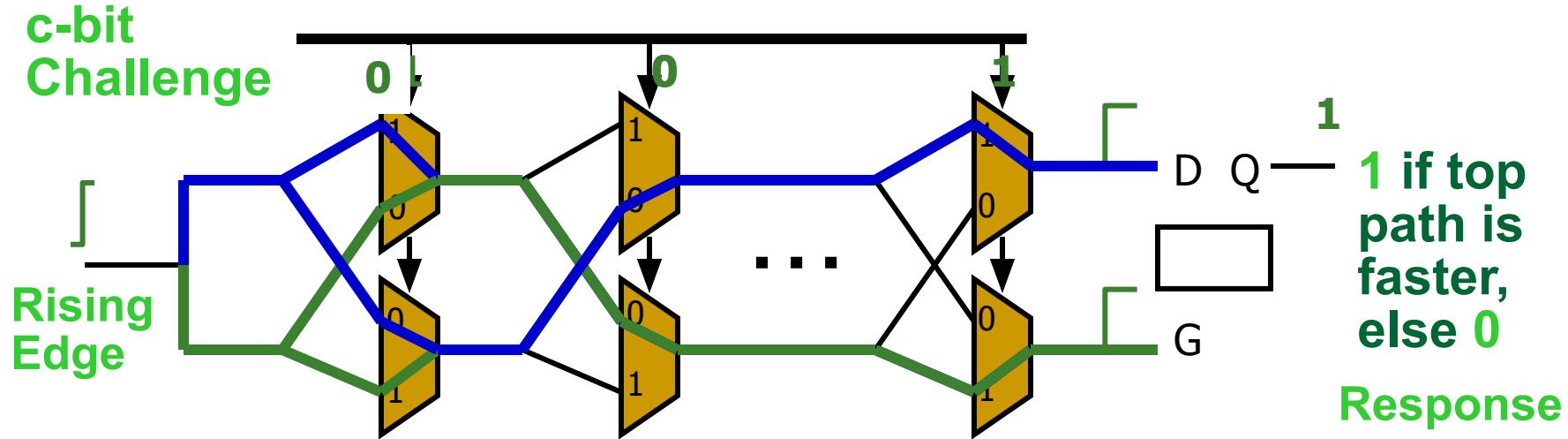
- Because of process variations, no two Integrated Circuits are identical
- Experiments in which *identical circuits with identical layouts* were placed on different FPGAs show that path delays vary enough across ICs to use them for identification.



Reminder... 2-Bit Multiplexer (Mux2)



A Candidate: Silicon PUF



- Compare two paths with an identical delay in design
 - Random process variation determines which path is faster
 - An arbiter outputs 1-bit digital response
- **Path delays in an IC are statistically distributed due to random manufacturing variations**

PUF Characteristics

Strong PUF: Significant, often unique state-space.
Challenge-response pairs unique or nearly unique.

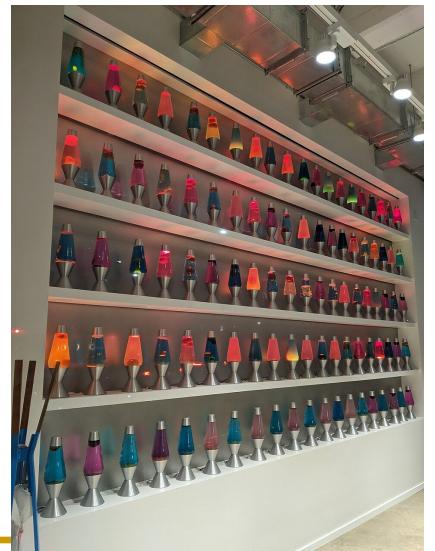
Weak PUF: Constrained State space, challenge-response pairs may not be unique.

Intrinsic PUF: Relies on naturally occurring differences in process variation.

Extrinsic PUF: Introduces entropy using additional carefully tuned dedicated circuits.

Implicit PUF: Rely on self-contained randomness within system.

Explicit PUF: Rely on Sensor data received outside of system.

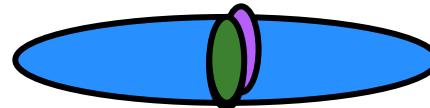


Experiments

- Fabricated candidate PUF on multiple ICs, 180nm TSMC
- Apply 100 random challenges and observe responses

100 bits of response

**At 70C measurement
noise for chip = 2 bits**



Baseline Measurement noise for Chip = 0.5

**Distance between Chip X and Y
responses = 24 bits**

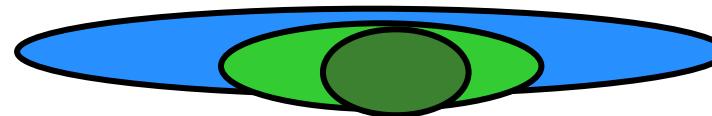


Measurement Attacks and Software Attacks

Can an adversary create a *software clone* of a given PUF chip?

**Distance between Chip X and Y
responses = 24**

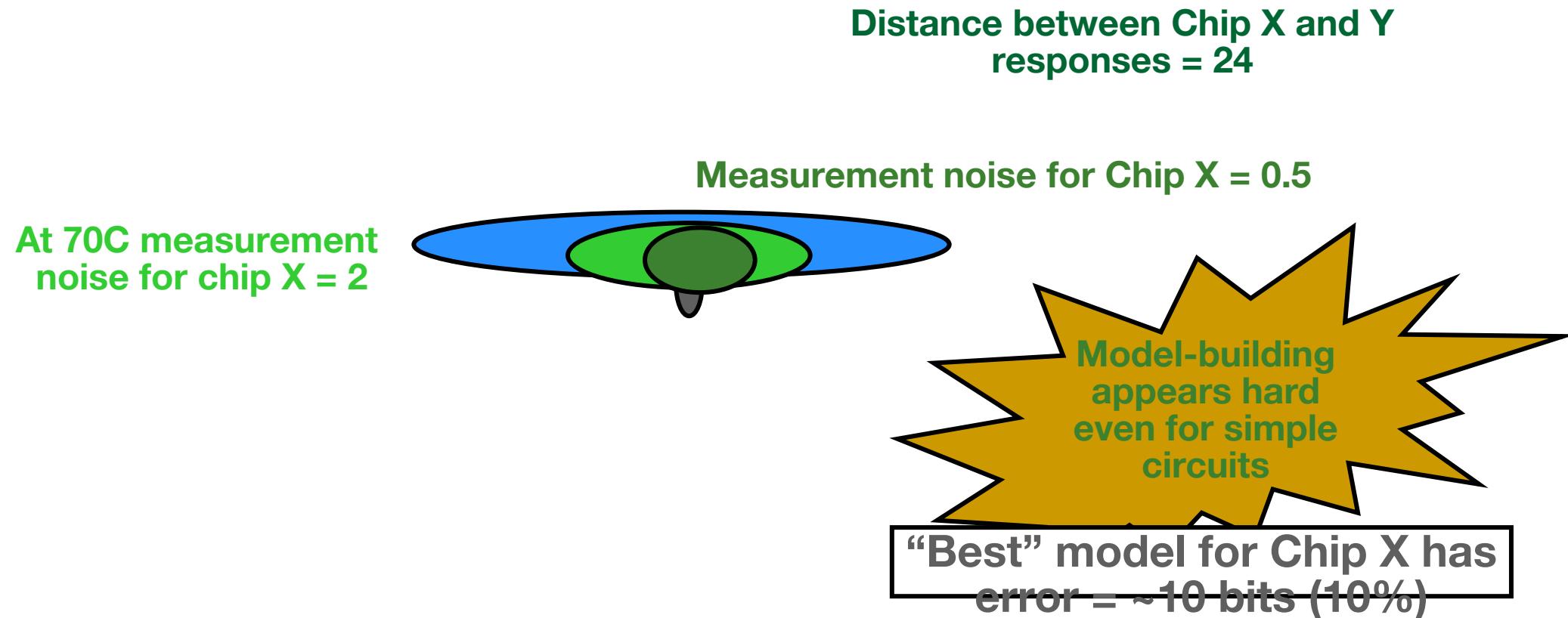
**At 70C measurement
noise for chip = 2**



Measurement noise for Chip = 0.5

Measurement Attacks and Software Attacks

Can an adversary create a *software clone* of a given PUF chip?



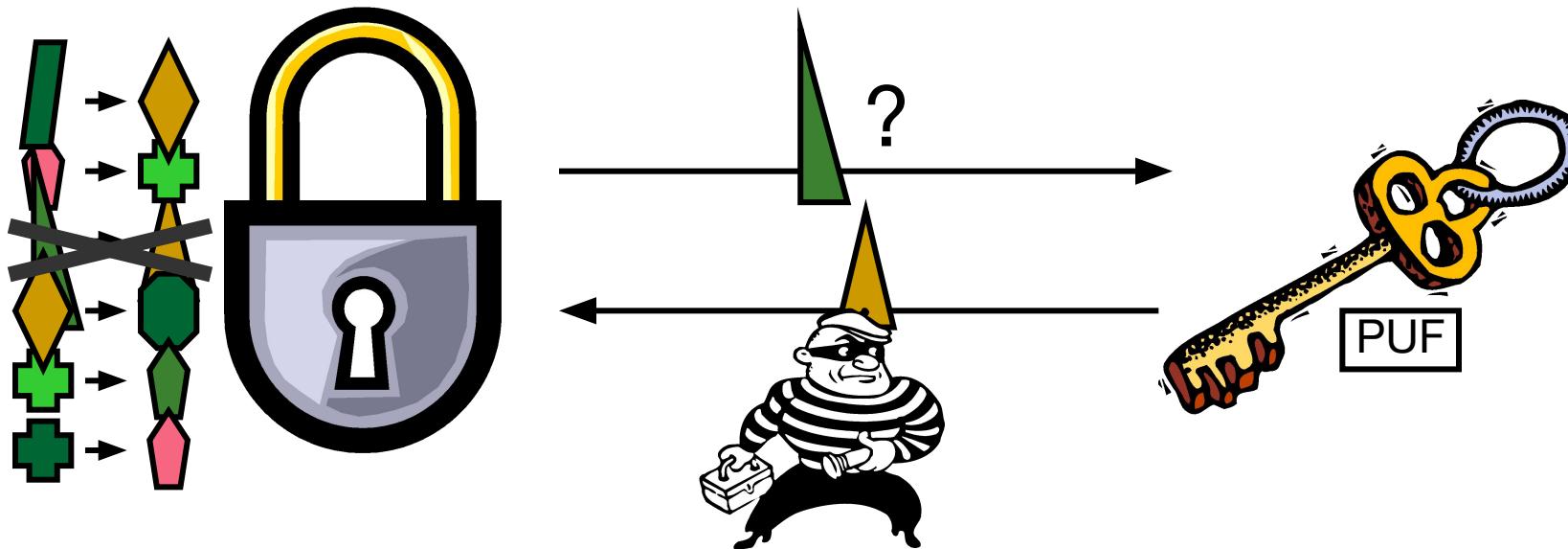
Physical Attacks

- Make PUF delays depend on overlaid metal layers and package
- Invasive attack (e.g., package removal) changes PUF delays and destroys PUF
- Non-invasive attacks are still possible
 - To find wire delays one needs to find precise relative timing of transient signals as opposed to looking for 0's and 1's
 - Wire delay is not a number but a function of challenge bits and adjacent wire voltages and capacitances

Using a PUF as an Unclonable Key

A Silicon PUF can be used as an unclonable key.

- The lock has a database of challenge-response pairs.
- To open the lock, the key has to show that it knows the response to one or more challenges.



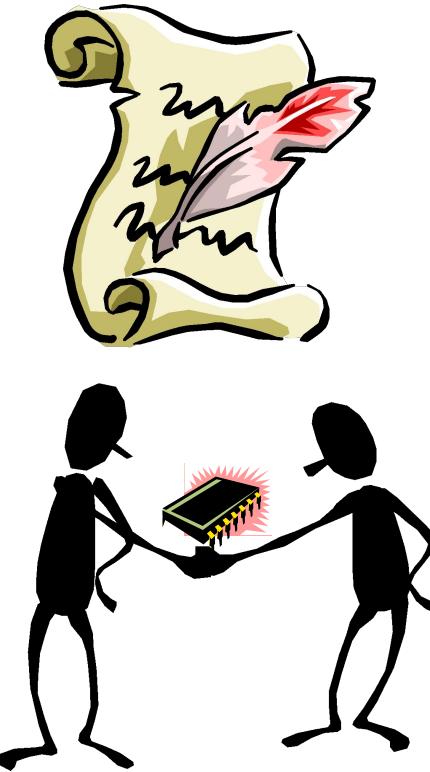
Applications

- **Anonymous Computation**

Alice wants to run computations on Bob's computer, and wants to make sure that she is getting correct results. A certificate is returned with her results to show that they were correctly executed.

- **Software Licensing**

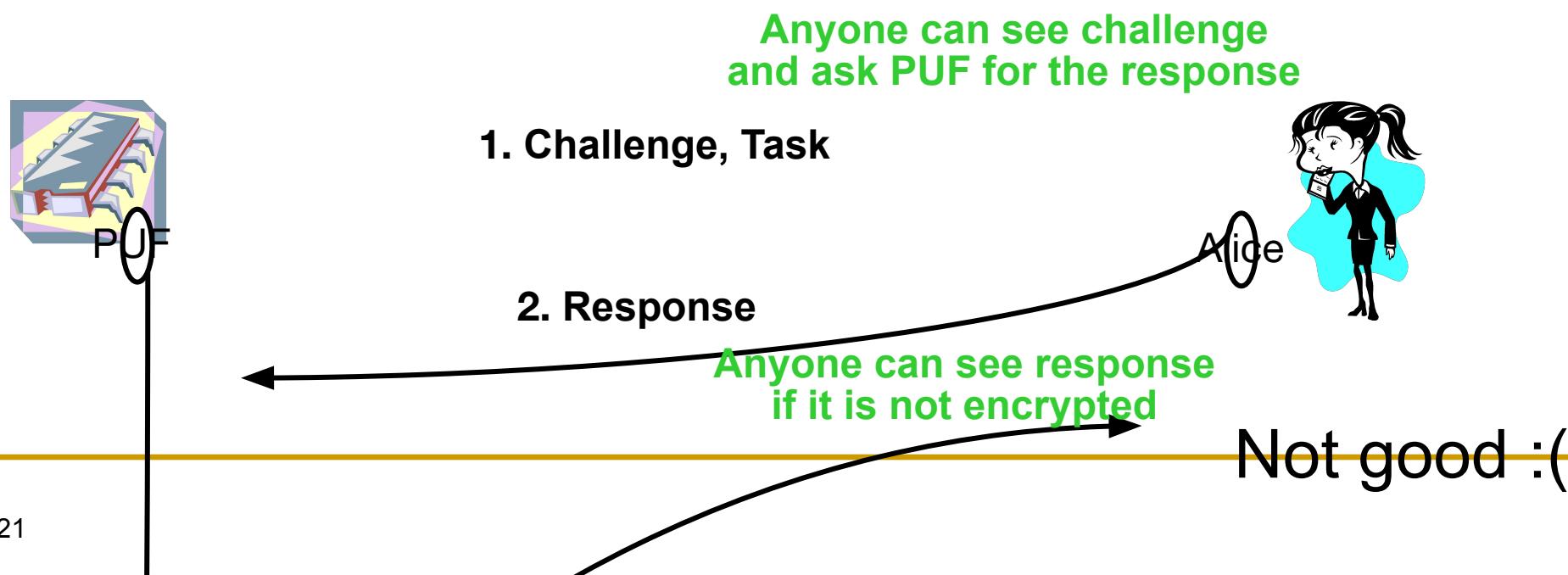
Alice wants to sell Bob a program which will only run on Bob's chip (identified by a PUF). The program is copy-protected so it will not run on any other chip.



We can enable the above applications by trusting only a single-chip processor that contains a silicon PUF.

Sharing a Secret with a Silicon PUF

Suppose Alice wishes to share a secret with the silicon PUF
She has a challenge response pair that no one else knows,
which can authenticate the PUF
She asks the PUF for the response to a challenge

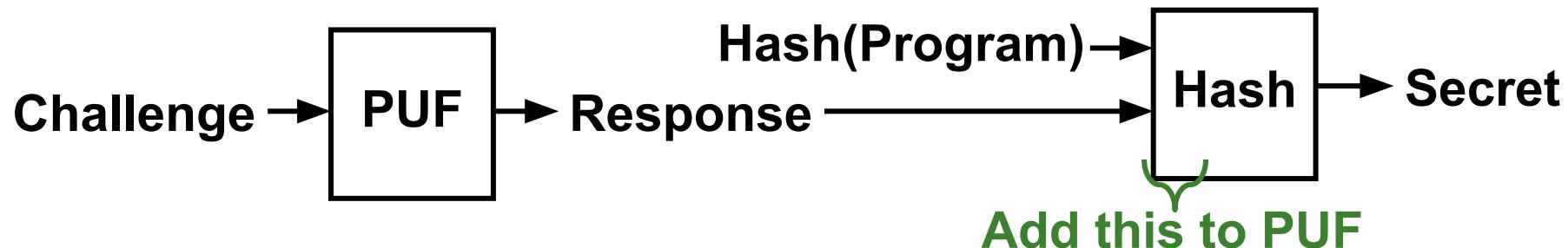


Restricting Access to the PUF

- To prevent the attack, the man in the middle must be **prevented** from finding out the response.
- **Alice's program** must be able to establish a shared secret with the PUF, **the attacker's program** must not be able to get the secret.

⇒ **Combine response with hash of program.**

- The PUF can only be accessed via the **GetSecret** function:



Cryptographic Hash Function

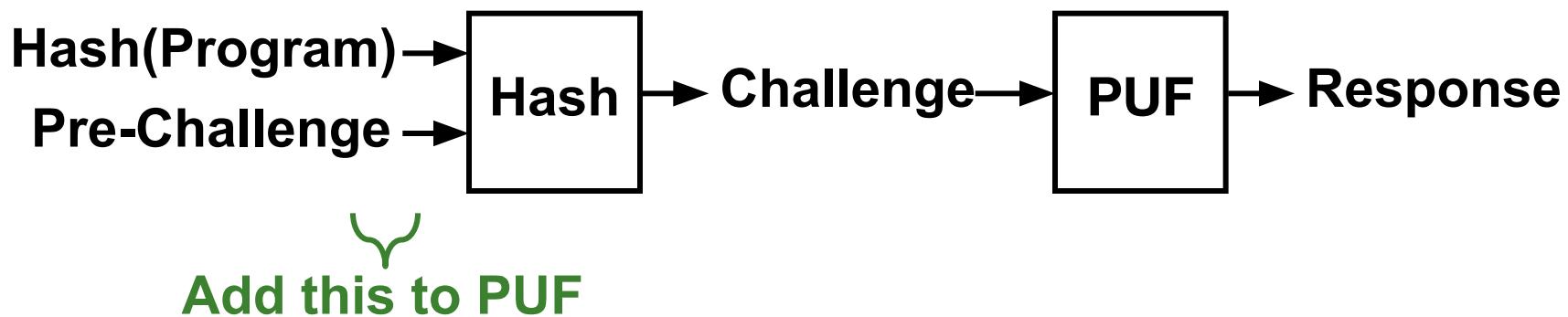
- Crypto hash function $h(x)$ must provide
 - **Compression** —output length is small
 - **Efficiency** — $h(x)$ easy to compute for any x
 - **One-way** —given a value y it is infeasible to find an x such that $h(x) = y$
 - **Weak collision resistance** — given x and $h(x)$, infeasible to find $y \neq x$ such that $h(y) = h(x)$
 - **Strong collision resistance** — infeasible to find any x and y , with $x \neq y$ such that $h(x) = h(y)$

Getting a Challenge-Response Pair

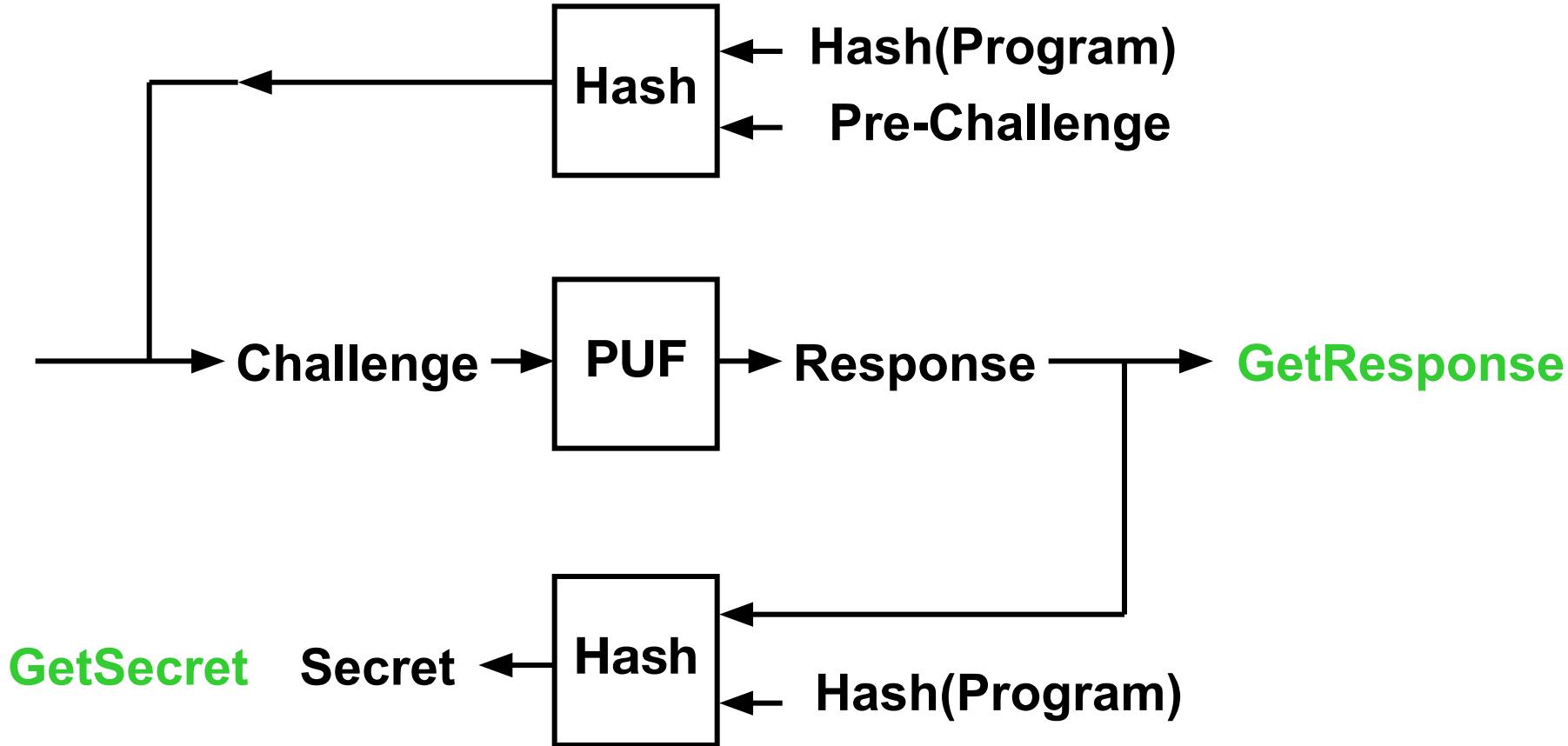
- Now Alice **can** use a Challenge-Response pair to generate a shared **secret** with the PUF equipped device.
- But Alice **can't** get a Challenge-Response pair in the first place since the PUF **never** releases responses directly.
→ **An extra function that can return responses is needed.**

Getting a Challenge-Response Pair – 2

- Let Alice use a Pre-Challenge.
 - Use program hash to prevent eavesdroppers from using the pre-challenge.
-
- The PUF has a GetResponse function



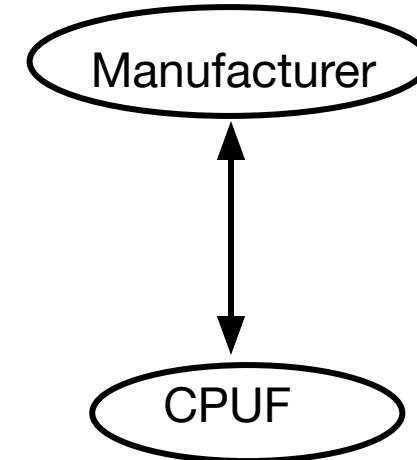
Controlled PUF Implementation



Challenge-Response Pair Management: Bootstrapping

When a Controlled PUF (CPUF) has just been produced, the manufacturer wants to generate a challenge-response pair.

1. Manufacturer provides Pre-challenge and Program.
2. CPUF produces Response.
3. Manufacturer gets Challenge by computing $\text{Hash}(\text{Hash}(\text{Program}), \text{PreChallenge})$.
4. Manufacturer has (Challenge, Response) pair where Challenge, Program, and $\text{Hash}(\text{Program})$ are public, but Response is not known to anyone since Pre-challenge is thrown away



Software Licensing

Program (EncCode, Challenge)

 Secret = GetSecret(Challenge)

 Code = Decrypt(EncCode, Secret) → Hash(Program)

 Run Code

EncCode has been encrypted with Secret by Manufacturer

Secret is known to the manufacturer because he knows Response to Challenge and can compute

 Secret = Hash(Hash(Program), Response)

Adversary cannot determine Secret because he does not know Response or Pre-Challenge

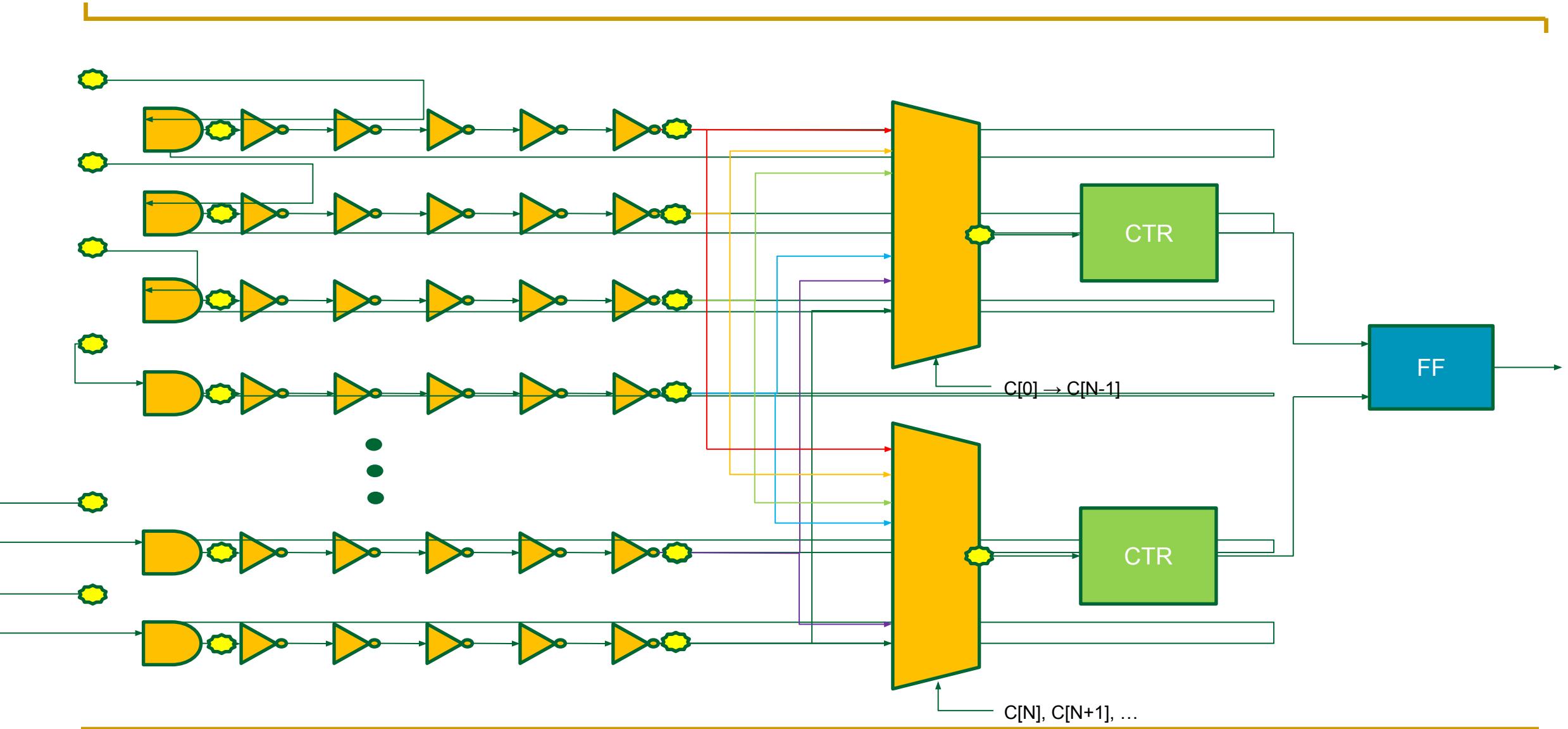
If adversary tries a different program, a different secret will be generated because Hash(Program) is different

Summary

- PUFs provide secret “key” and CPUFs enable sharing a secret with a hardware device
 - CPUFs are not susceptible to model-building attack if we assume physical attacks cannot discover the PUF response
 - Control protects PUF by obfuscating response, and PUF protects the control from attacks by “covering up” the control logic
 - Shared secrets are volatile
-

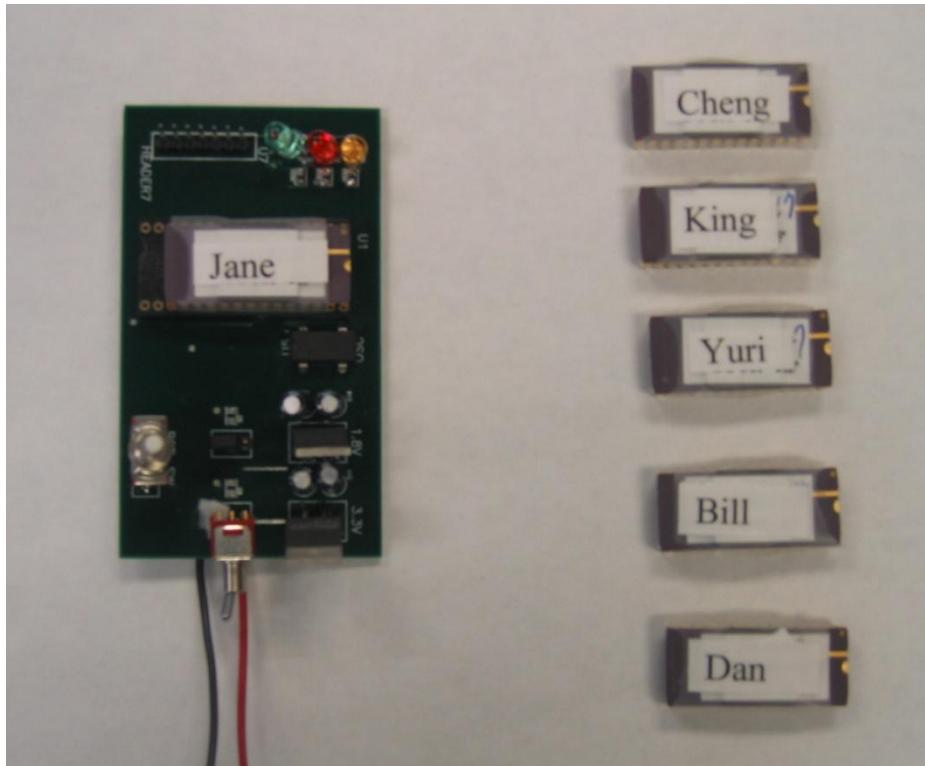
More on Physically Unclonable Functions (PUFs)

Ring Oscillator (RO) PUF



PUF Experiments

- Fabricated 200 “identical” chips with PUFs in TSMC 0.18 μ on 5 different wafer runs



Security

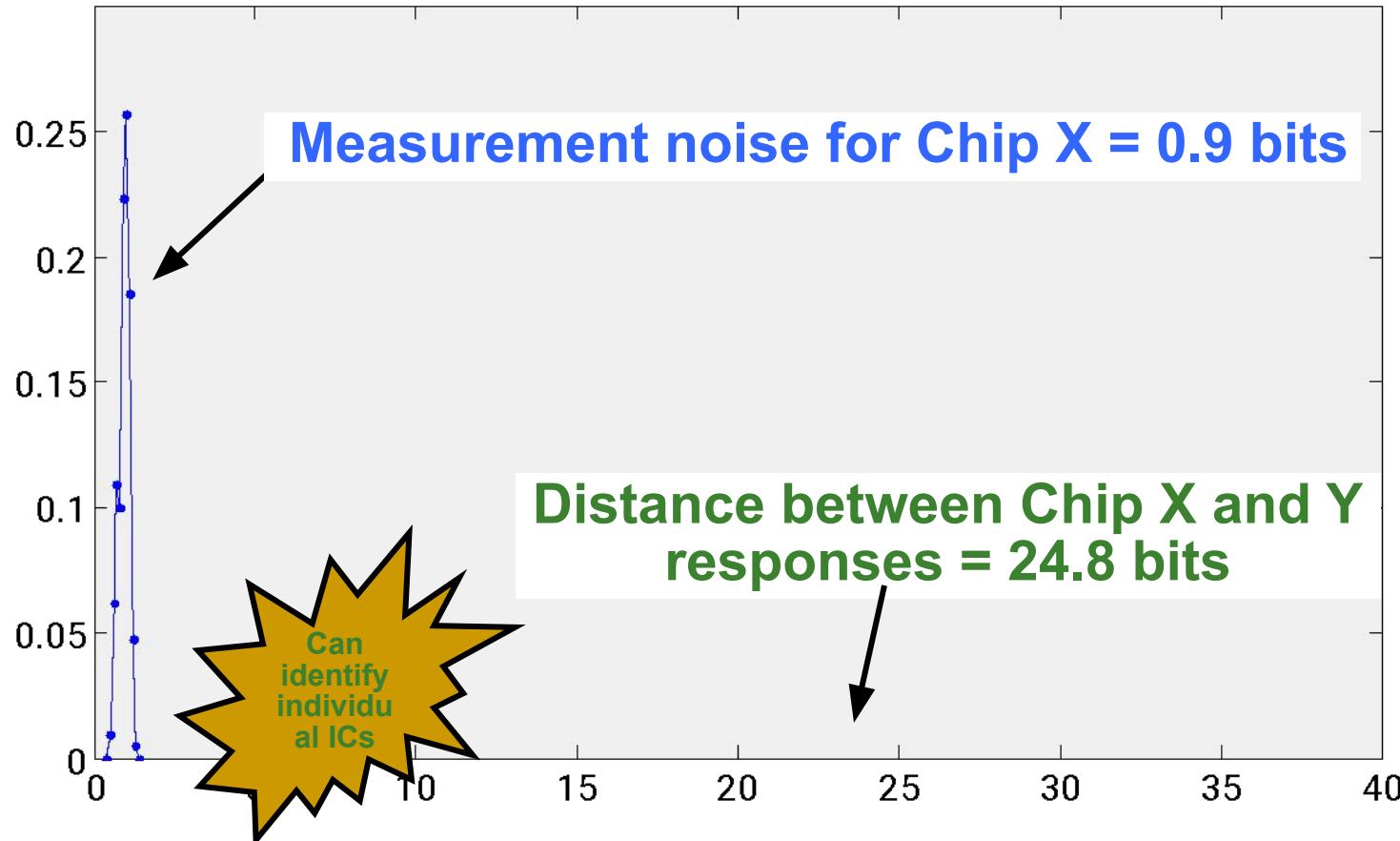
- What is the probability that a challenge produces **different responses** on two different PUFs?

Reliability

- What is the probability that a PUF output for a challenge **changes with temperature**?
 - With voltage variation?

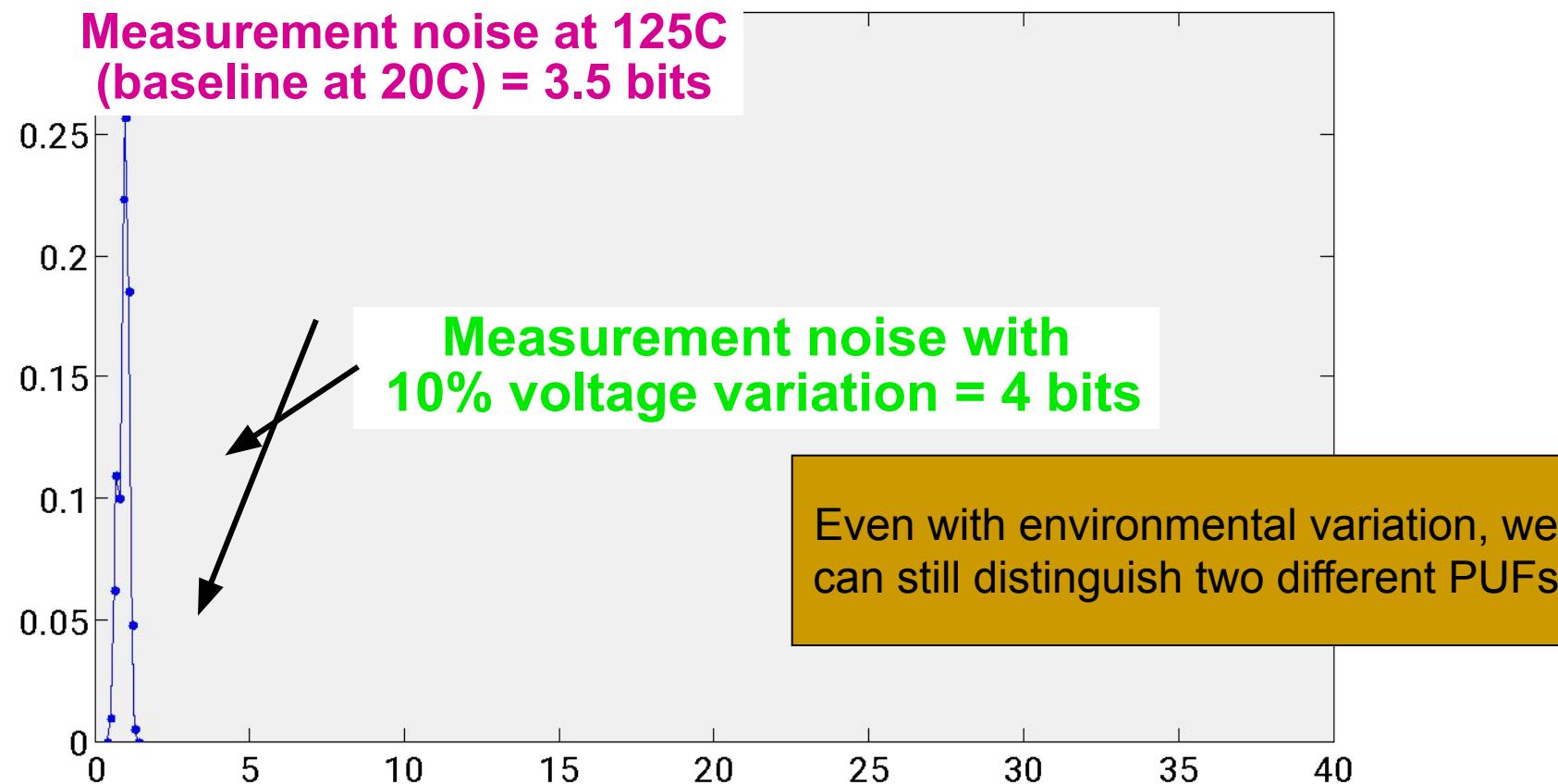
Inter-Chip Variation

- Apply random challenges and observe 100 response bits



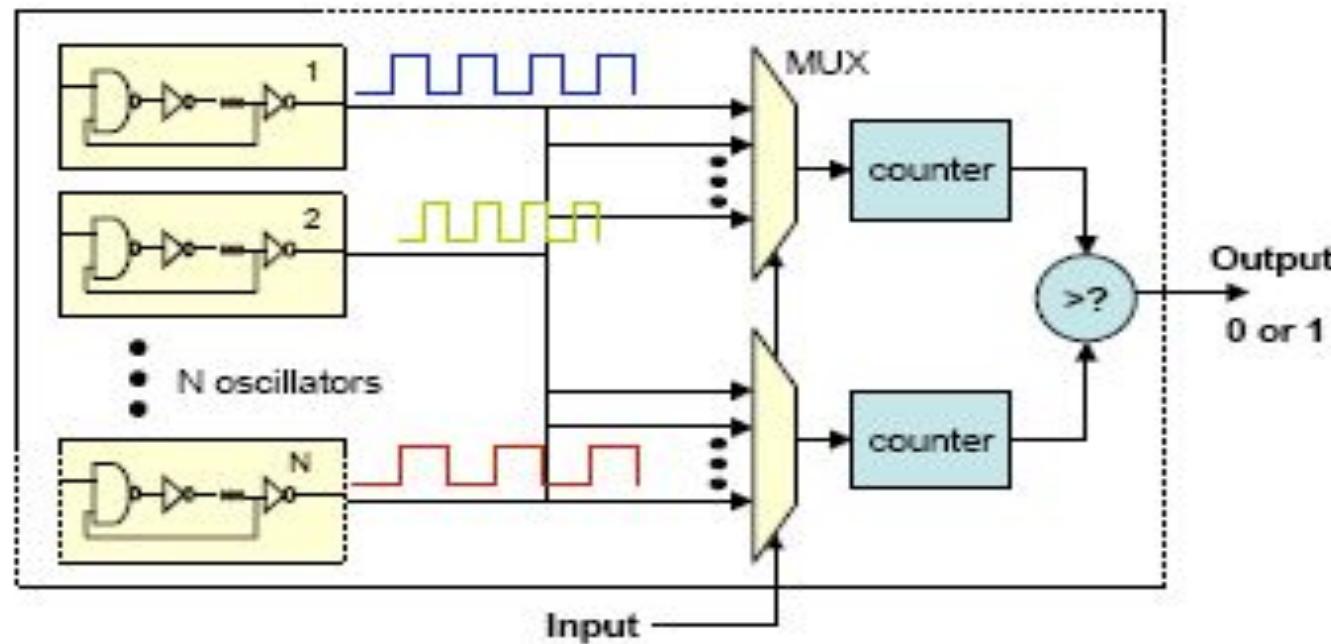
Environmental Variations

- What happens if we change voltage and temperature?



Ring-Oscillator (RO) PUF

- The structure relies on delay loops and counters instead of MUX and arbiters
- Better results on FPGA – more stable

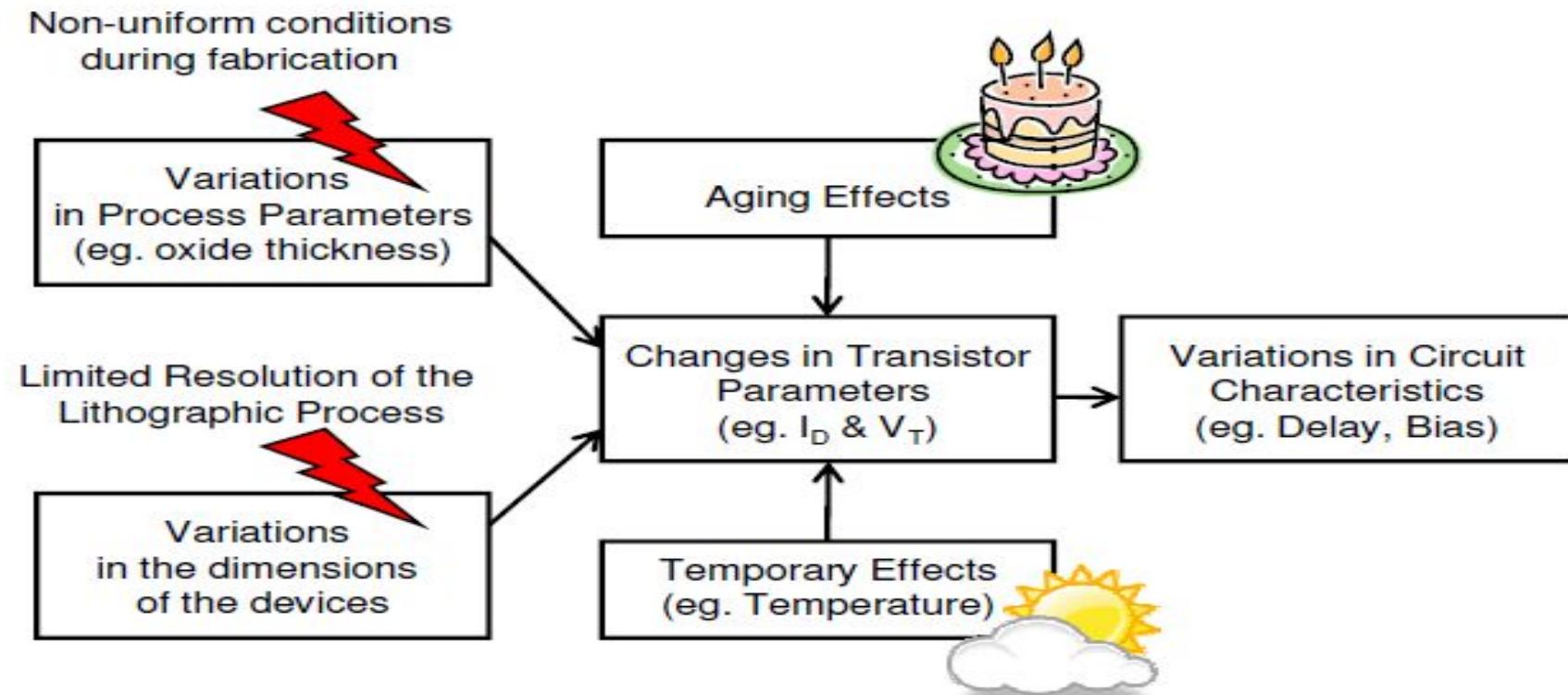


RO PUFs (cont'd)

- Easy to duplicate a ring oscillator and make sure the oscillators are identical
 - Much easier than ensuring the racing paths with equal path segments
- How many bits can we generate from the scheme in the previous page?
 - There are $N(N-1)/2$ distinct pairs, but the entropy is significantly smaller: $\log_2(N!)$
 - E.g., 35 ROs can produce 133 bits, 128 ROs can produce 716, and 1024 ROs can produce 8769

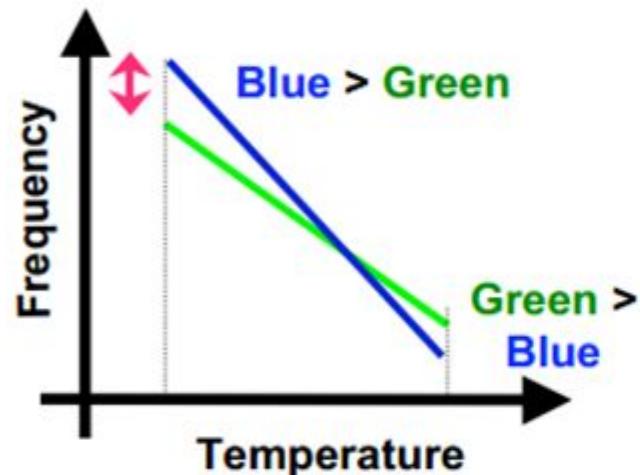
Reliability of RO PUFs

- Two types of reliability issues:
 - Aging:
 - Negative Bias Temperature Instability
 - Hot Carrier Injection (HCI)
 - Temp Dependent Dielectric Breakdown
 - Interconnect Failure
 - Temperature
 - Slows down the device
- PUFs are delicate and sensitive. Building reliable PUFs is difficult and \$\$\$.

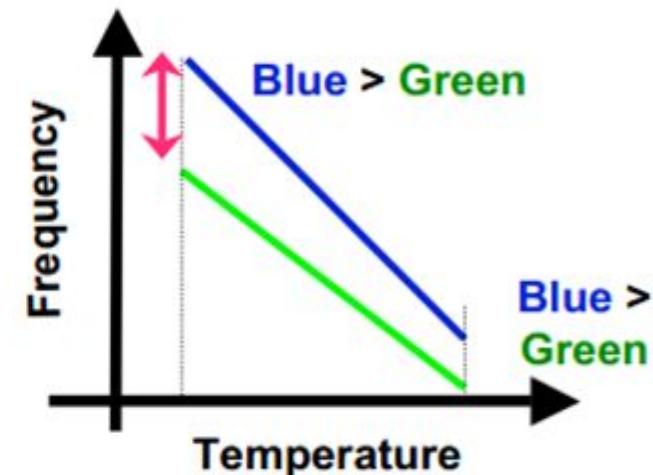


Reliability Enhancement

- Environmental changes have a large impact on the freq. (and even relative ones)



(a) Frequencies are close



(b) Frequencies are far apart

RO PUFs

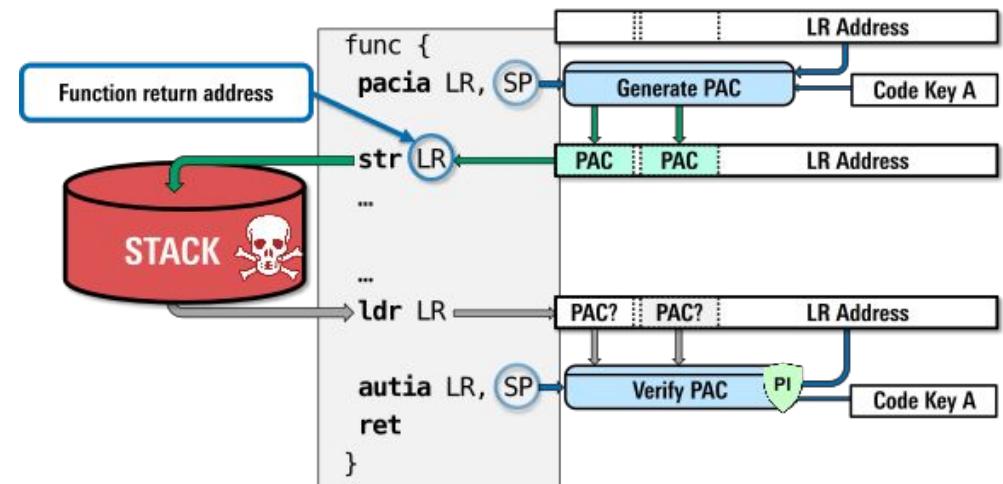
- ROs whose frequencies are far are more stable than the ones with closer frequencies
 - Possible advantage: do not use all pairs, but only the stable ones
 - It is easy to watch the distance in the counter and pick the very different ones.
 - Can be done during enrollment
 - RO PUF allows an easier implementation for both ASICs and FPGAs.
 - The **Arbiter** PUF is appropriate for resource constrained platforms such as RFIDs and the **RO PUF** is better for use in FPGAs and in secure processor design.
-

Side Quest!

Unpatchable Hardware flaw in Apple Silicon!

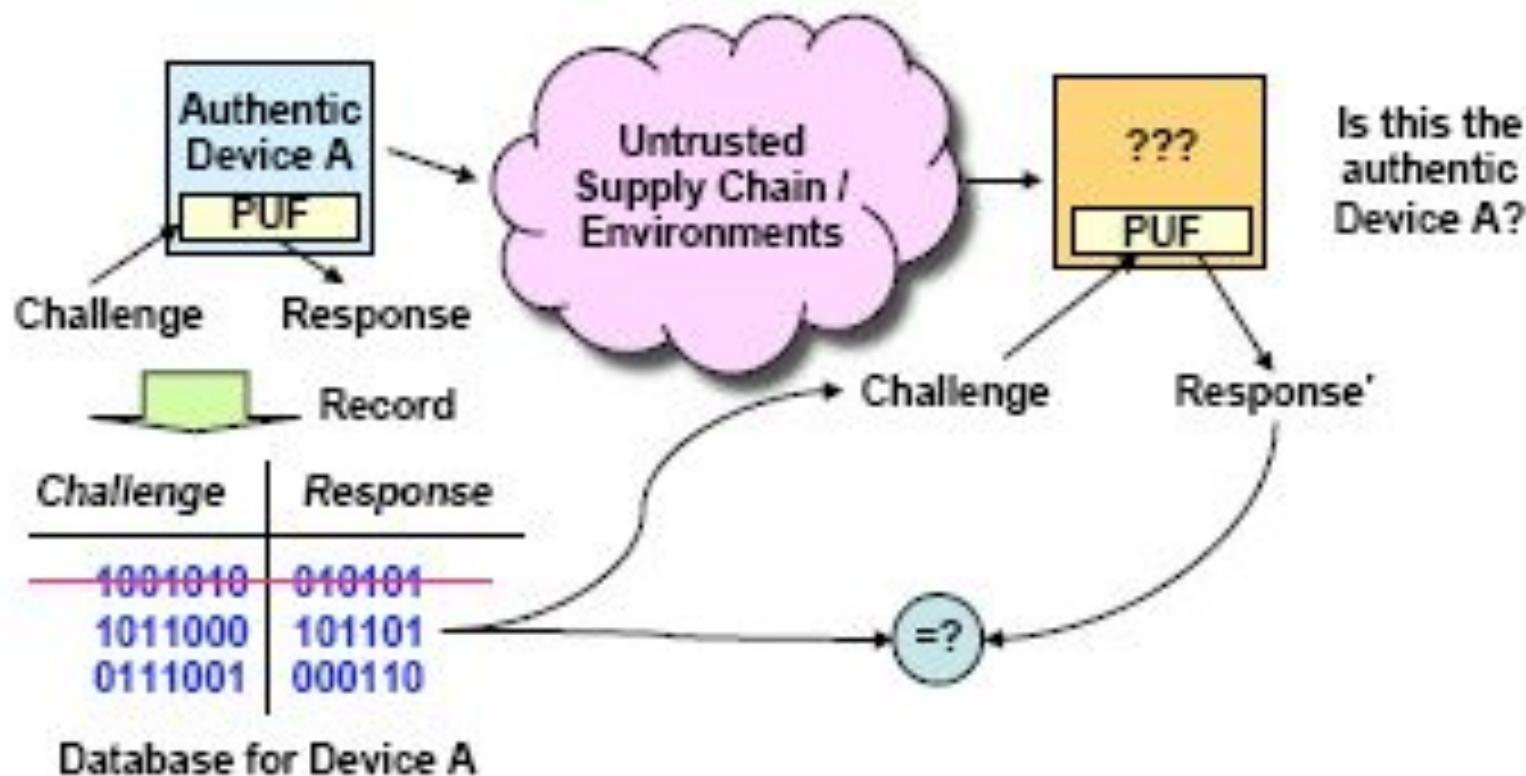
PACMAN -

<https://thehackernews.com/2022/06/mit-researchers-discover-new-flaw-in.html>



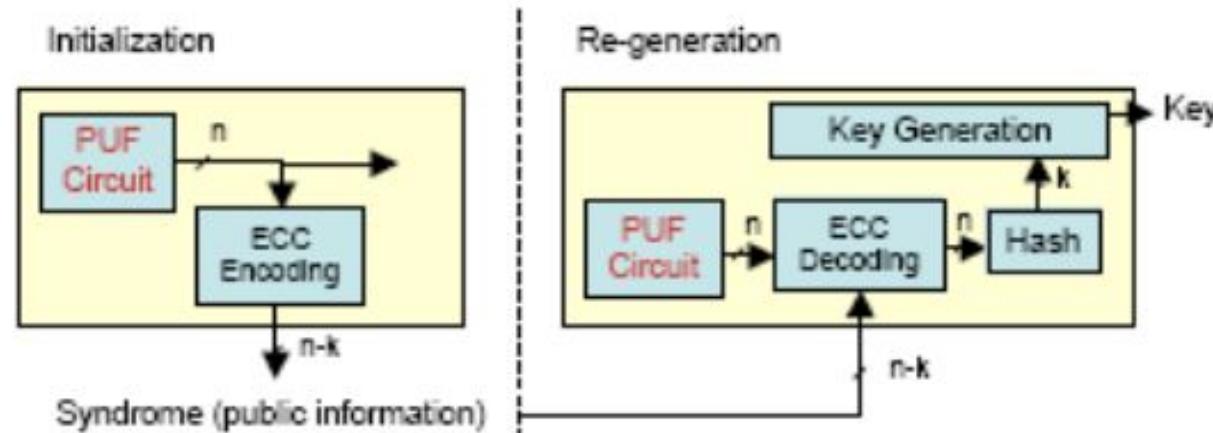
Applications – Authentication

- Same Challenges should not be used to prevent the man-in-the-middle attacks



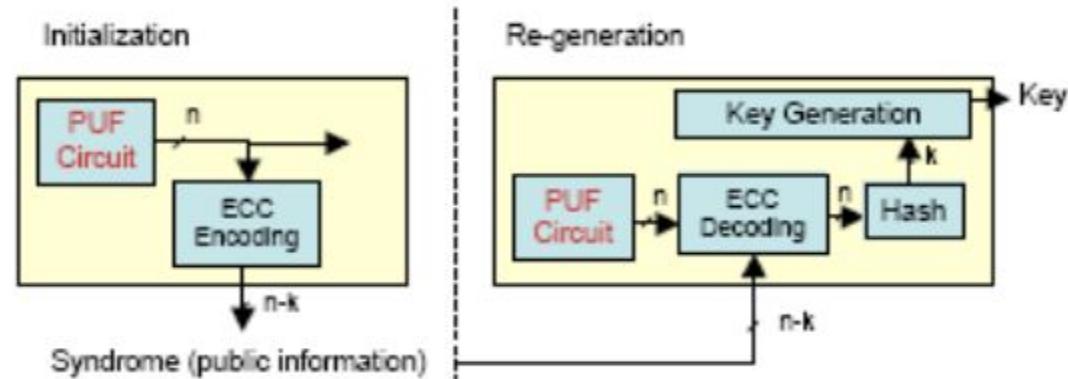
Application – Cryptographic Key Generation

- The instability is a problem
- Some crypto protocols (e.g., RSA) require specific mathematical properties that random numbers generated by PUFs do not have
- How can we use PUFs to generate crypto keys?
 - Error correction process: initialization and regeneration
 - There should be a one-way function that can generate the key from the PUF output



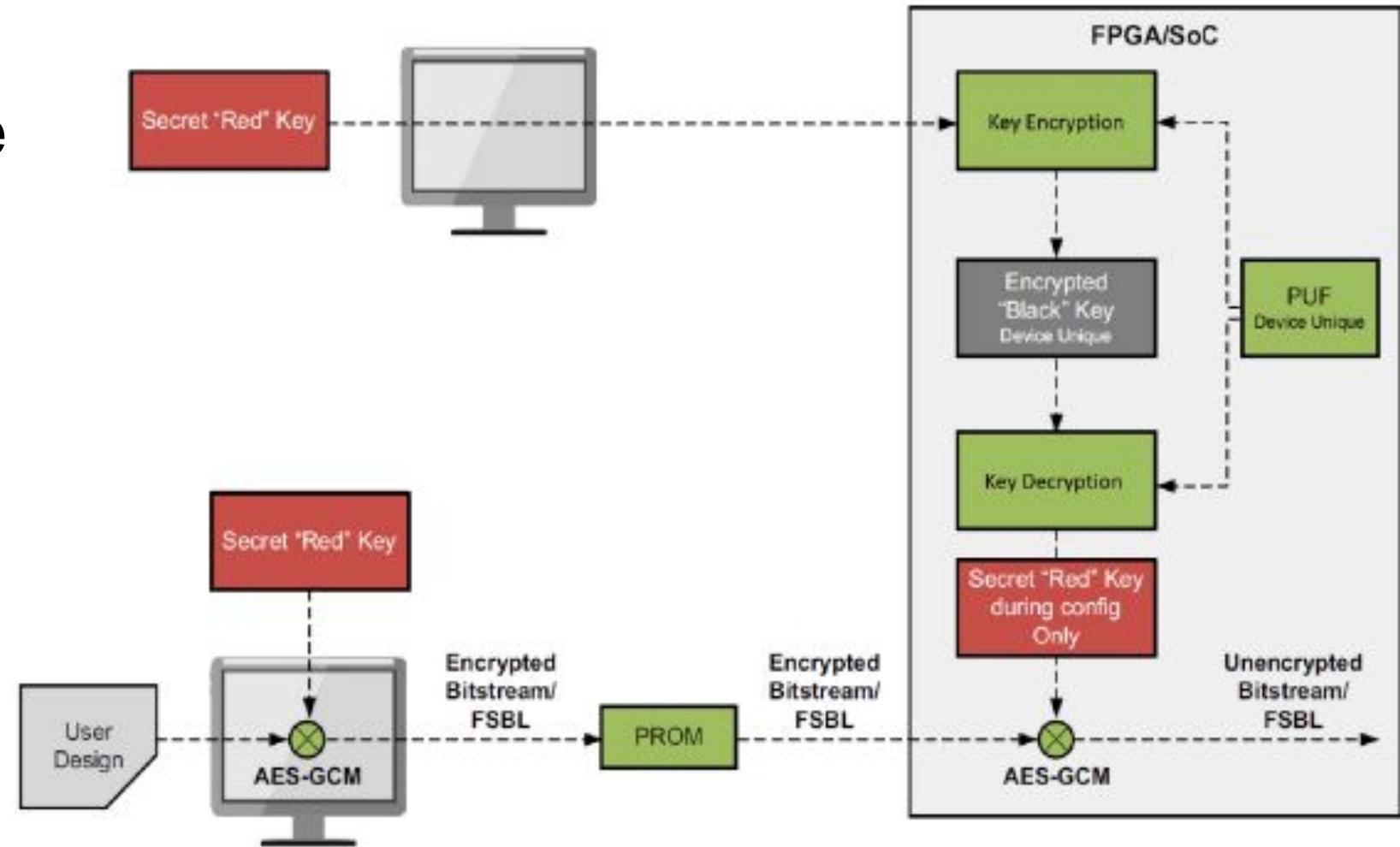
Crypto Key Generation

- Initialization: a PUF output is generated and error correcting code computes the syndrome (public info)
- Regeneration: PUF uses the syndrome from the initial phase to correct changes in the output
- Clearly, the syndrome reveals information about the circuit output and introduces vulnerabilities



Red/Black Keys

Use PUFs to hide
the on-chip key!



Experiments with RO PUFs

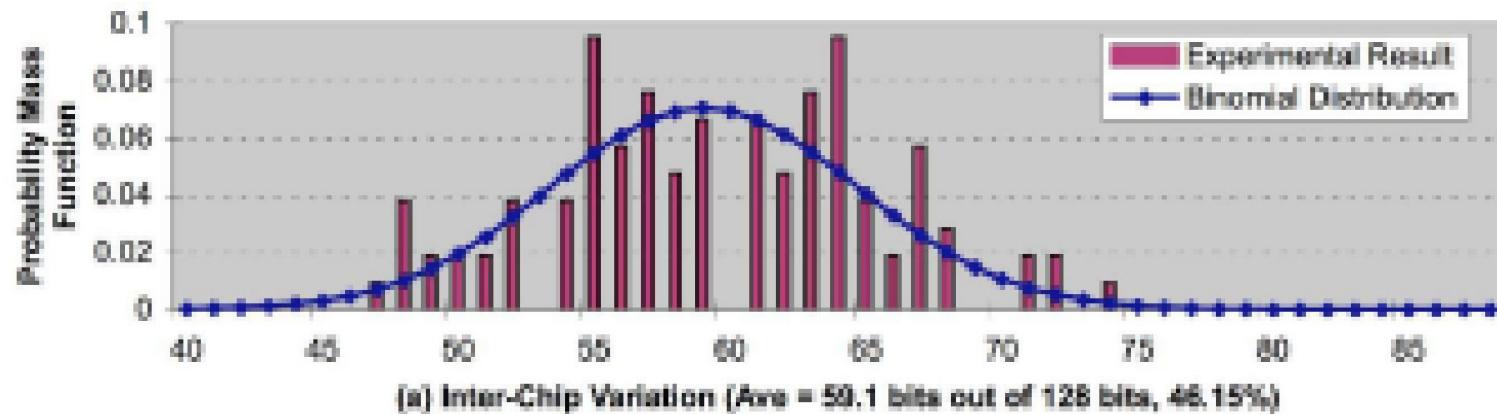
- Experiments done on 15 Xilinx Virtex4 LX25 FPGA (90nm)
 - They placed 1024 ROs in each FPGA as a 16-by-64 array
 - Each RO consisted of 5 INVs and 1 AND, implemented using look-up tables
 - The goal is to know if the PUF outputs are unique (**for security**) and reproducible (**for reliability and security**)
-

Reliability and Security Metrics

- *Inter-chip variation:* How many PUF output bits are different between PUF A and PUF B? This is a measure of uniqueness. If the PUF produces uniformly distributed independent random bits, the inter-chip variation should be 50% on average.
- *Intra-chip (environmental) variation:* How many PUF output bits change when re-generated again from a single PUF with or without environmental changes? This indicates the *reproducibility* of the PUF outputs. Ideally, the intra-chip variation should be 0%.

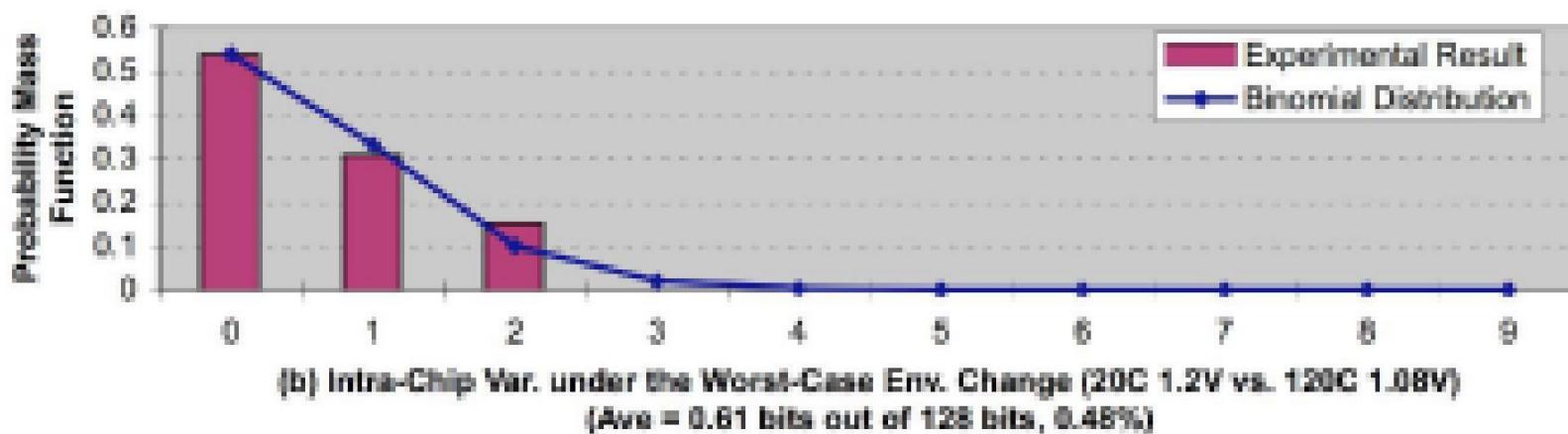
The Probability Distribution for Inter-chip Variations

- 128 bits are produced from each PUF
- x-axis: number of PUF o/p bits different b/w two FPGAs;
y-axis: probability
- Purple bars show the results from 105 pair-wise comparisons
- Blue lines show a binomial distribution with fitted parameters
($n=128$, $p = 0.4615$)
- Average inter-chip variations $0.4615 \sim 0.5$



The Probability Distribution for Intra-chip Variations

- PUF responses are generated at two different conditions and compared
- Changing the temperature from 20°C to 120°C and the core voltage from 1.2 to 1.08 altered the PUF o/p by ~0.6 bits (0.47%)
- Intra-chip variations is much lower than inter-chip – the PUF o/p did not change from small to moderate



Configurable Ring Oscillator

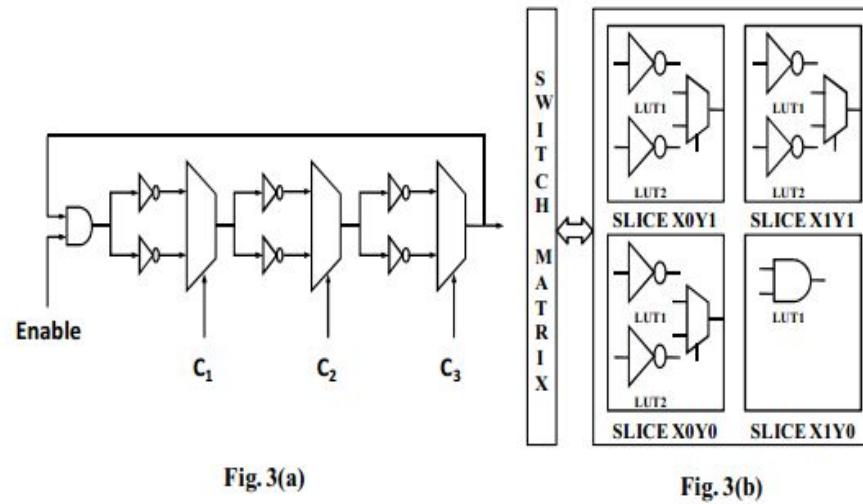
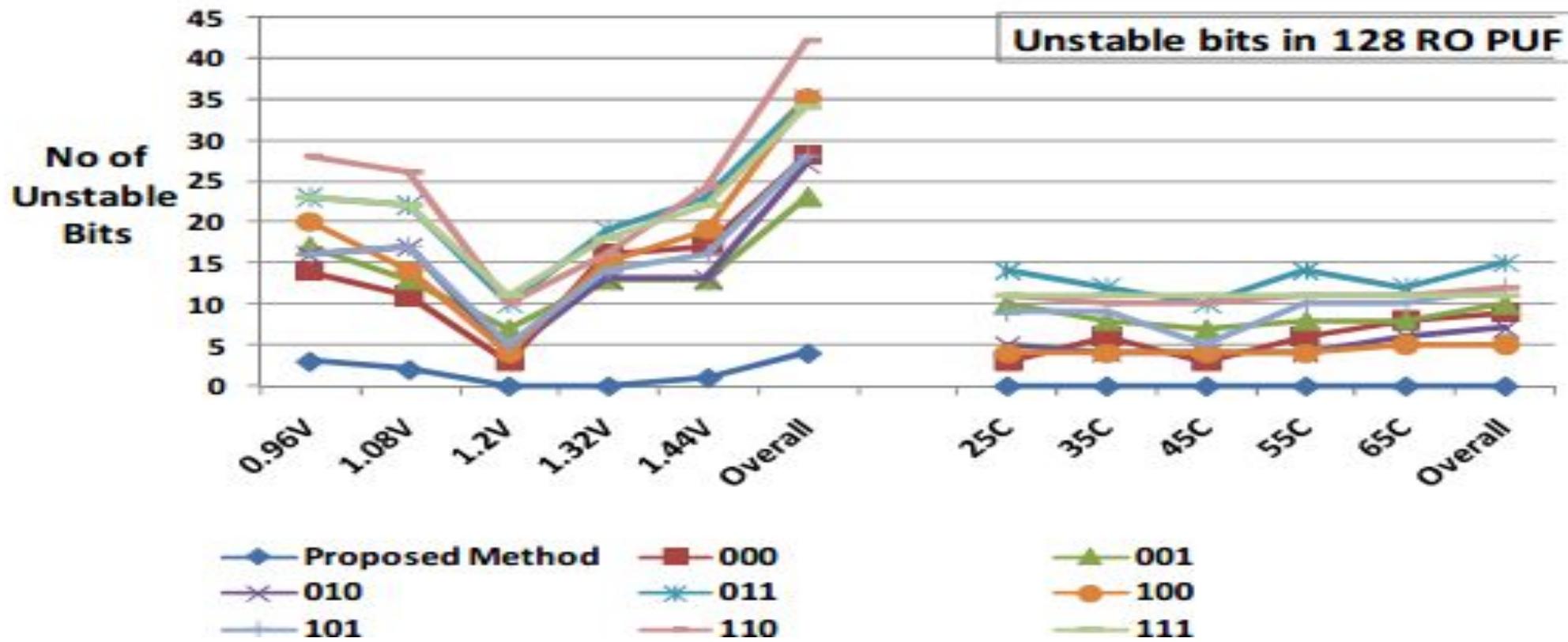


Table 1. Frequency differences in a configurable RO pair

$c_1c_2c_3$	Frequency of ROs in CLB i	Frequency of ROs in CLB j	Δf
000	f_0	f'_0	$ f_0 - f'_0 $
001	f_1	f'_1	$ f_1 - f'_1 $
010	f_2	f'_2	$ f_2 - f'_2 $
011	f_3	f'_3	$ f_3 - f'_3 $
100	f_4	f'_4	$ f_4 - f'_4 $
101	f_5	f'_5	$ f_5 - f'_5 $
110	f_6	f'_6	$ f_6 - f'_6 $
111	f_7	f'_7	$ f_7 - f'_7 $

- The pair which has the maximum difference in frequency in CRO is selected.

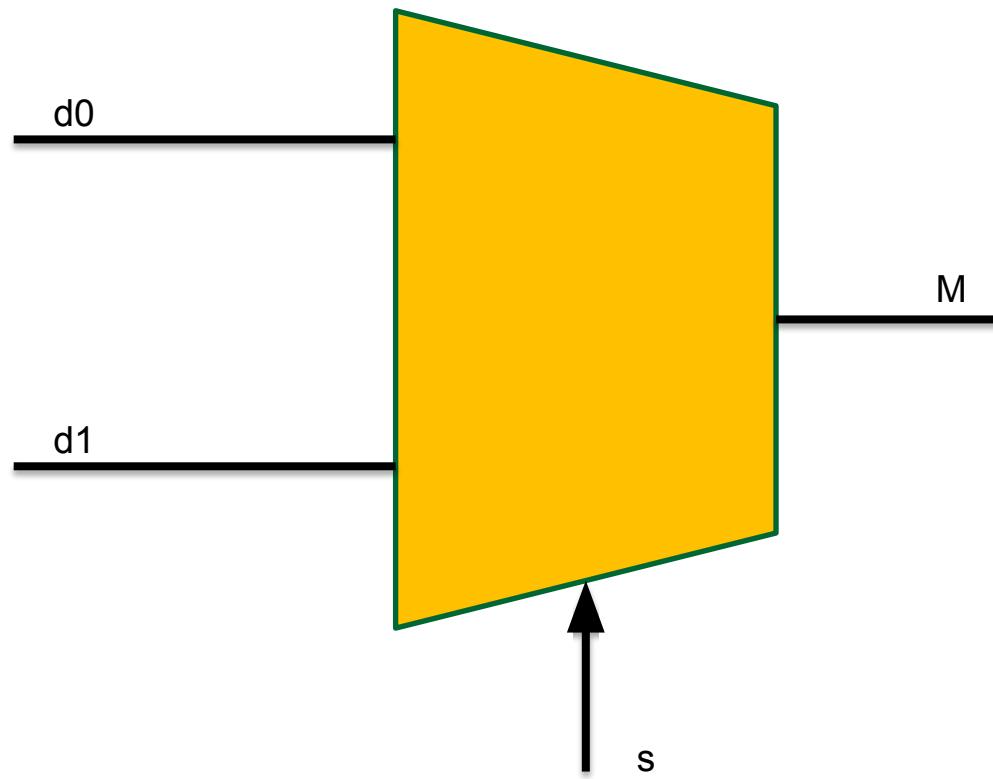
Configurable Ring Oscillator



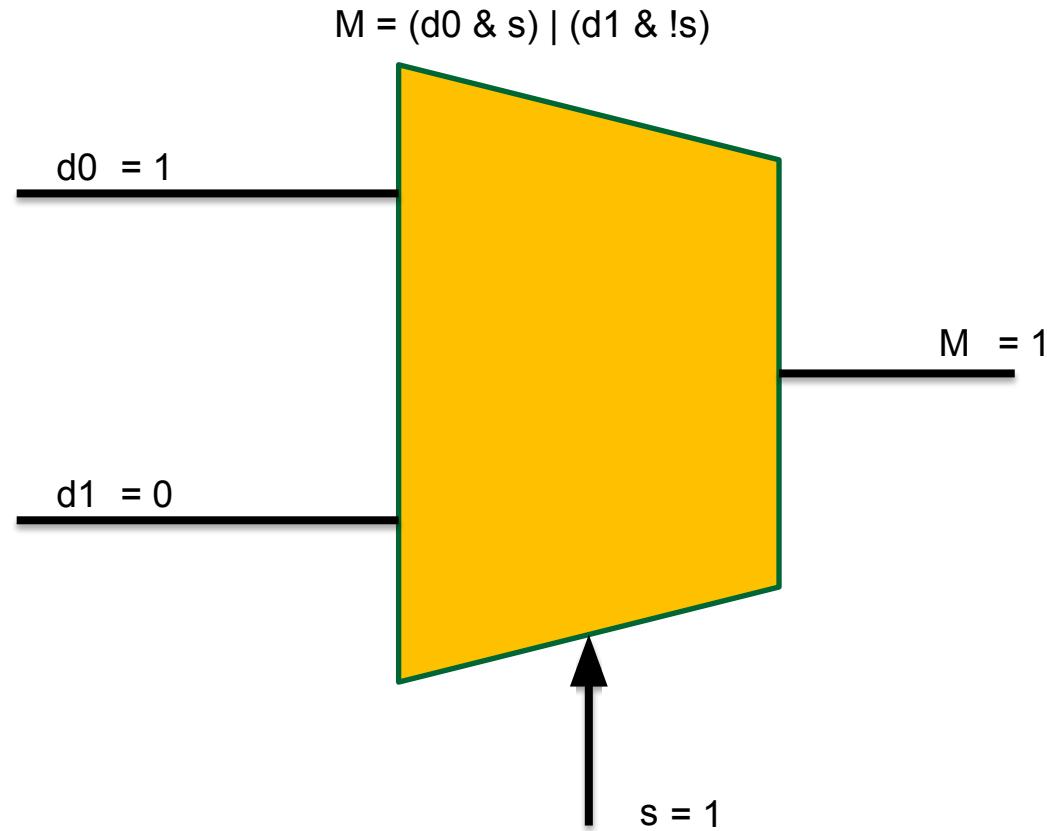
Introduction

- PUF – Physically Unclonable Function
 - RO – Ring Oscillator
 - MUX – Multiplexer
 - SRAM – Static Random Access Memory
 - LUT – Look Up Table
 - FF – Flip Flop
-
- Types of PUFs:
 - Arbiter PUF – Signal races between MUXes
 - RO PUF – Signal generated from ROs to trigger counter
 - Butterfly PUF – Exciting a cross-coupled circuit
-

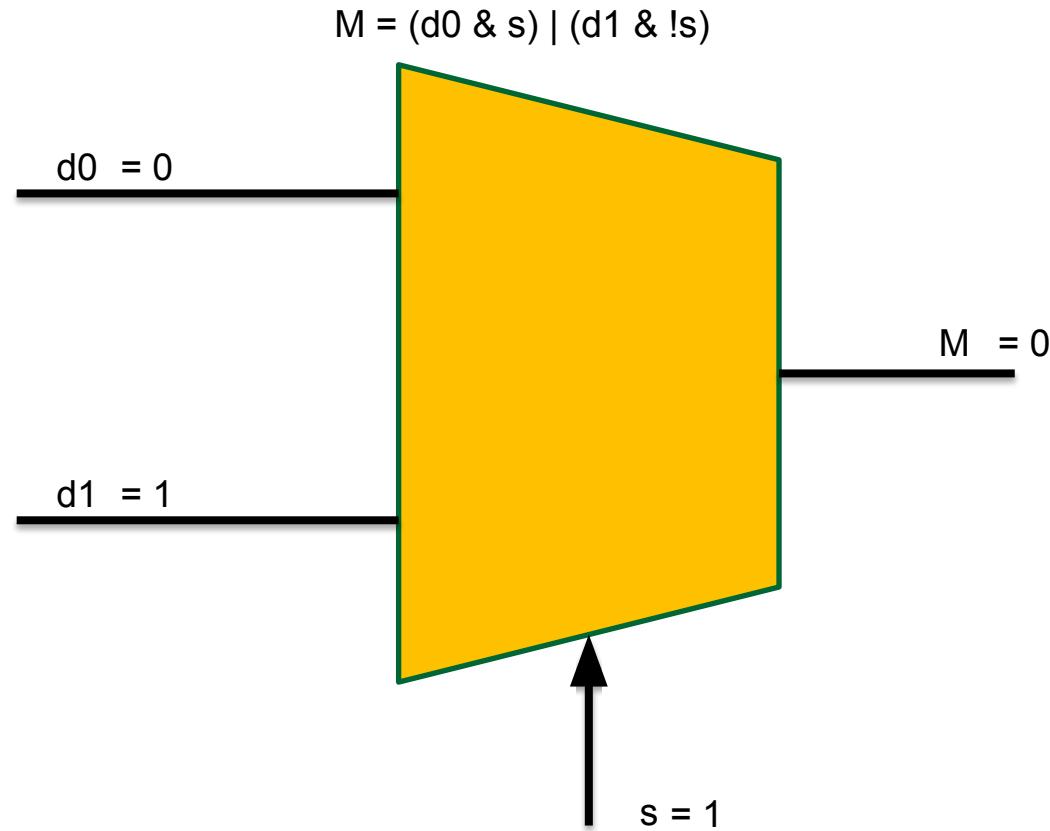
The Arbiter PUF



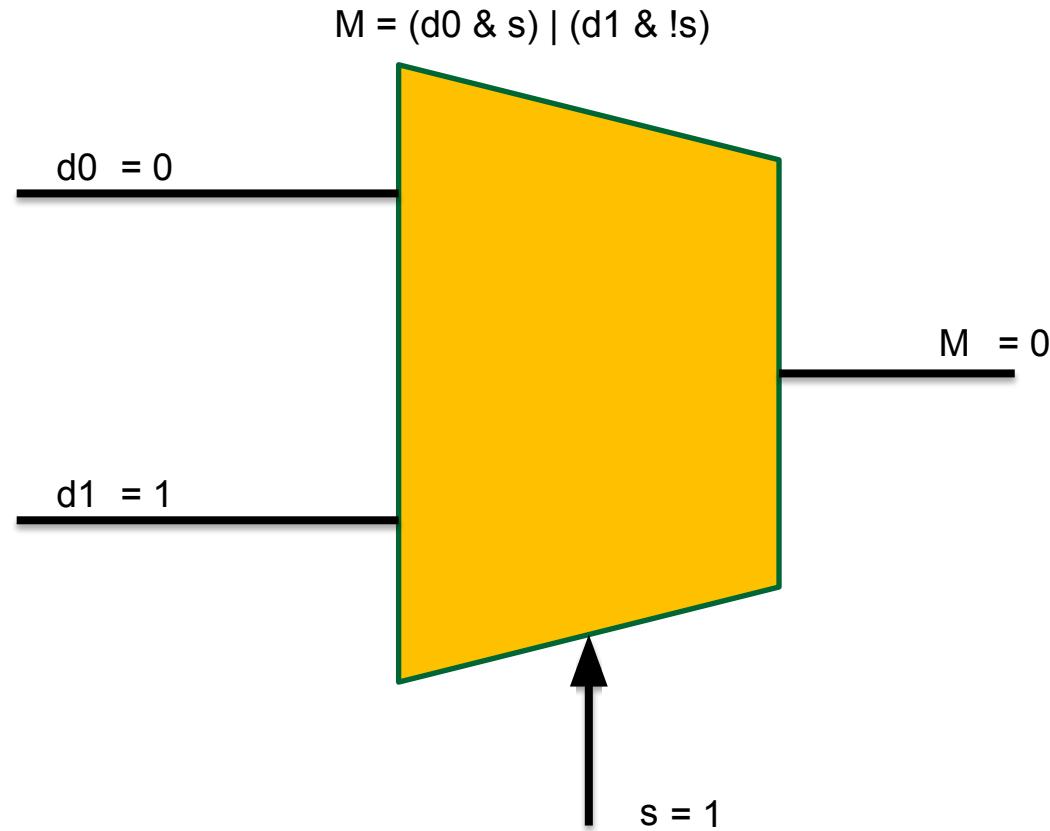
The Arbiter PUF



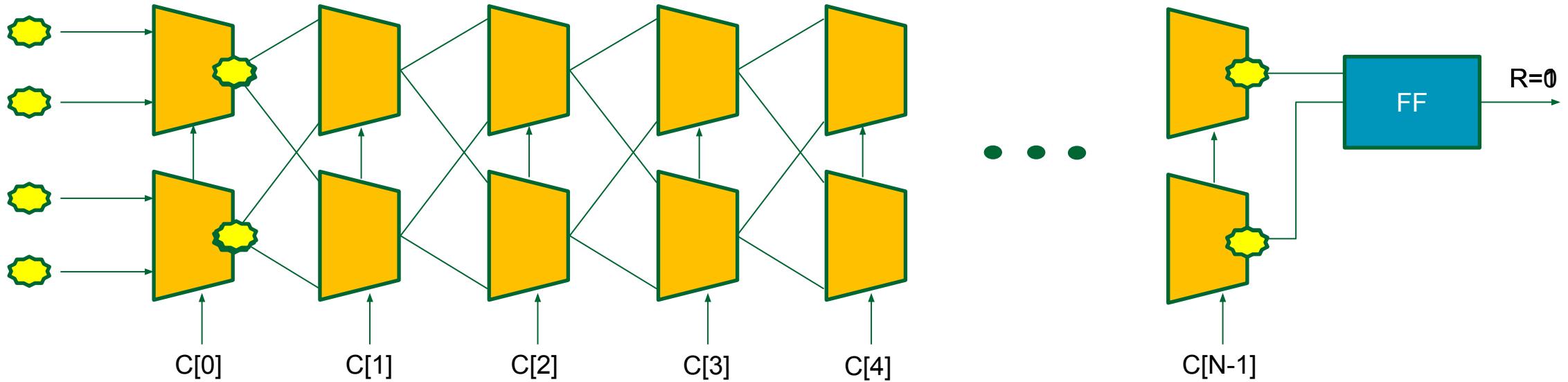
The Arbiter PUF



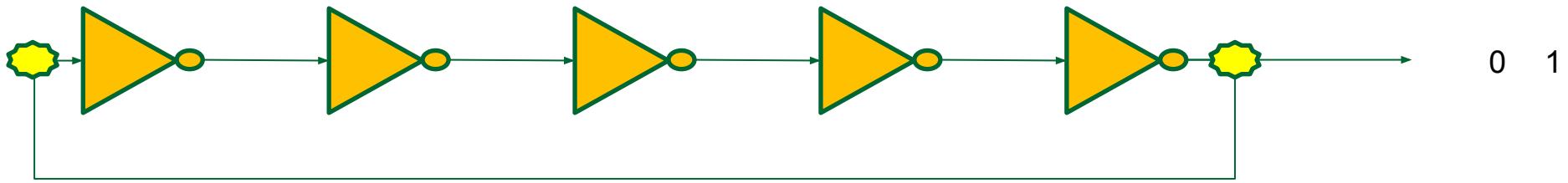
The Arbiter PUF



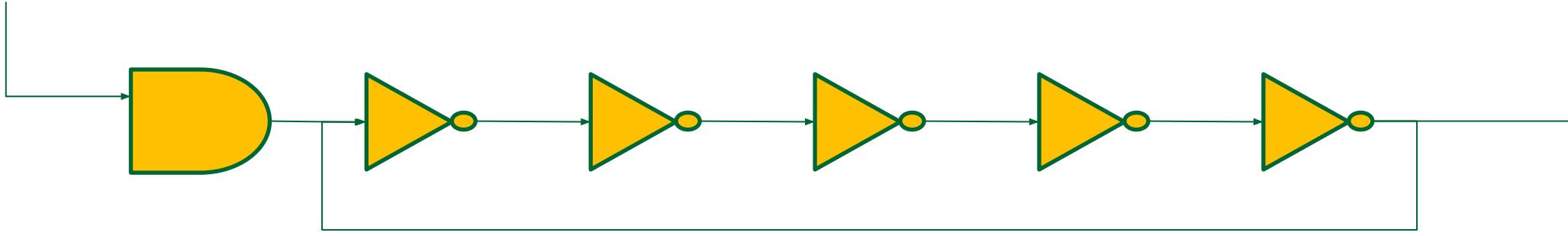
The Arbiter PUF



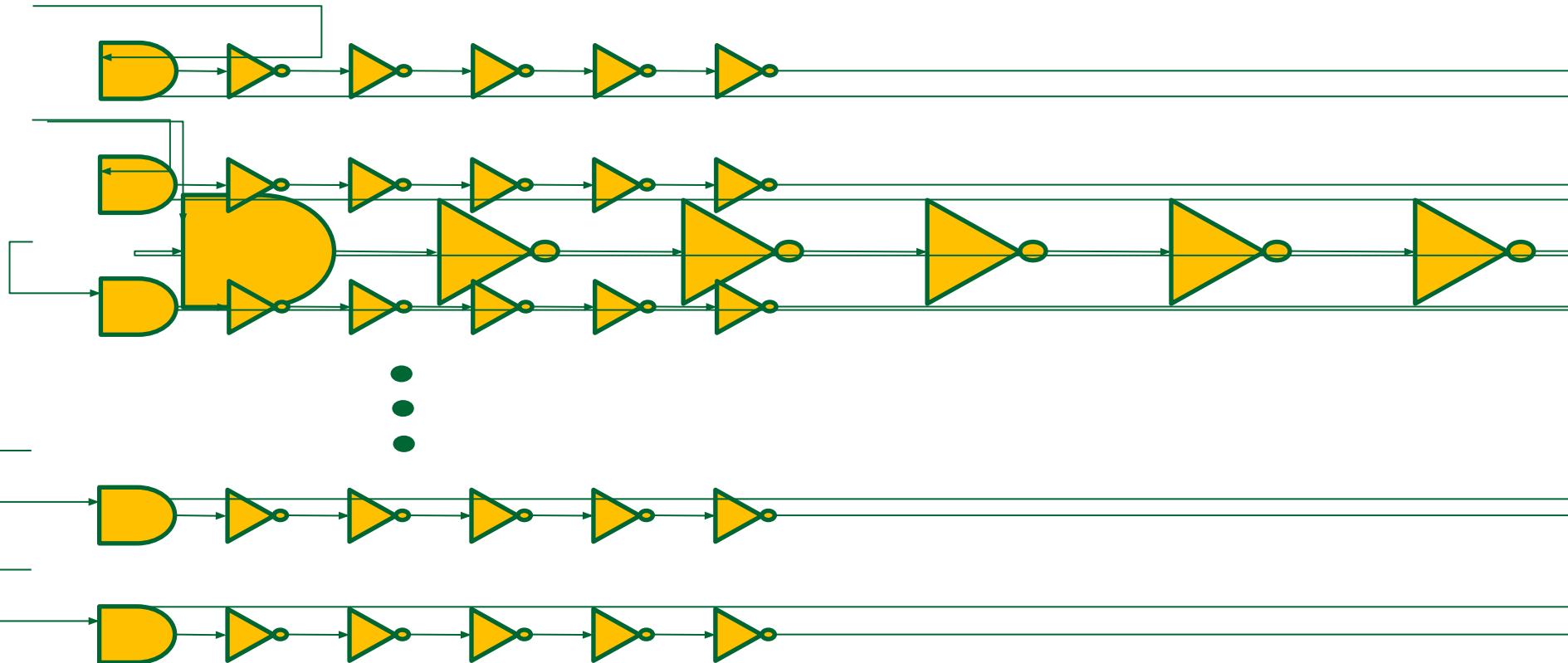
Ring Oscillator (RO) PUF



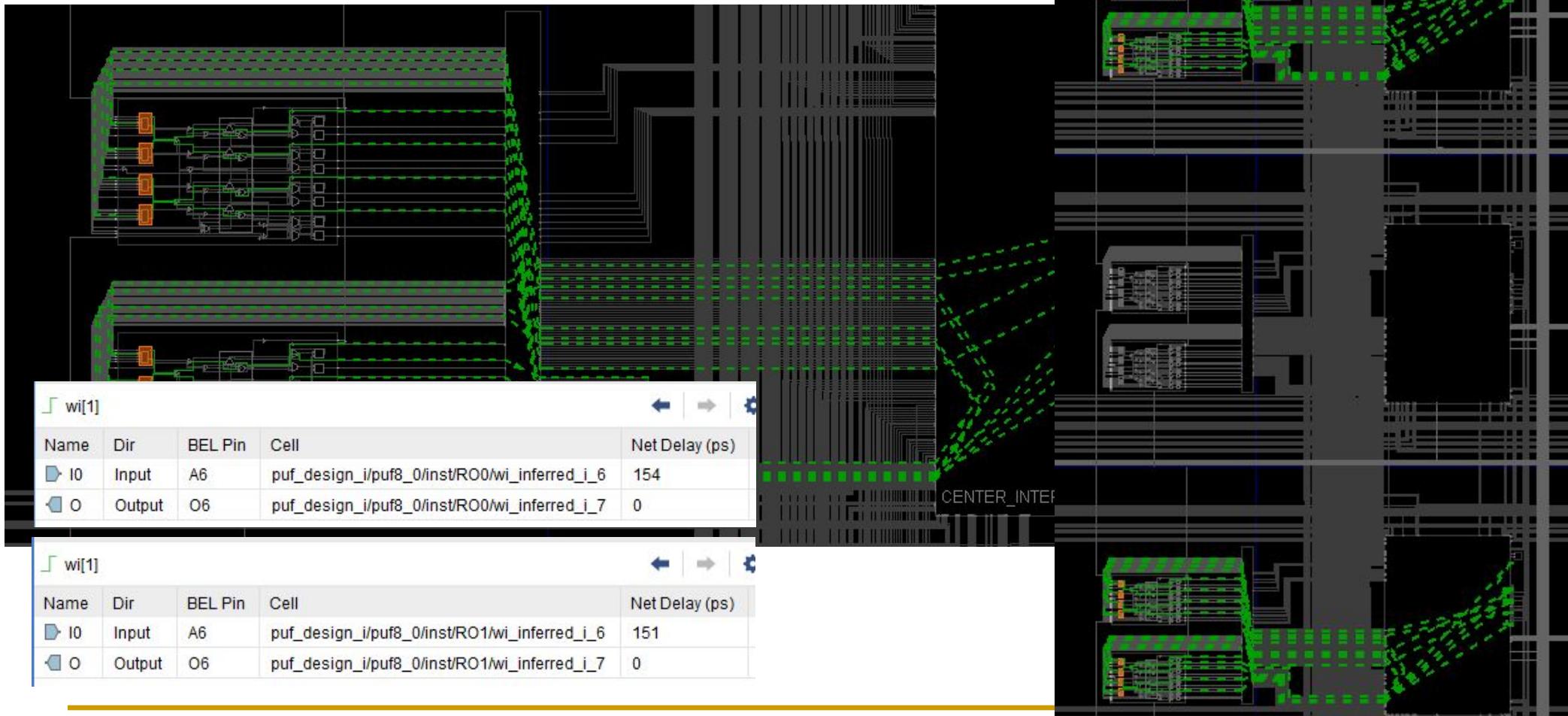
Ring Oscillator (RO) PUF



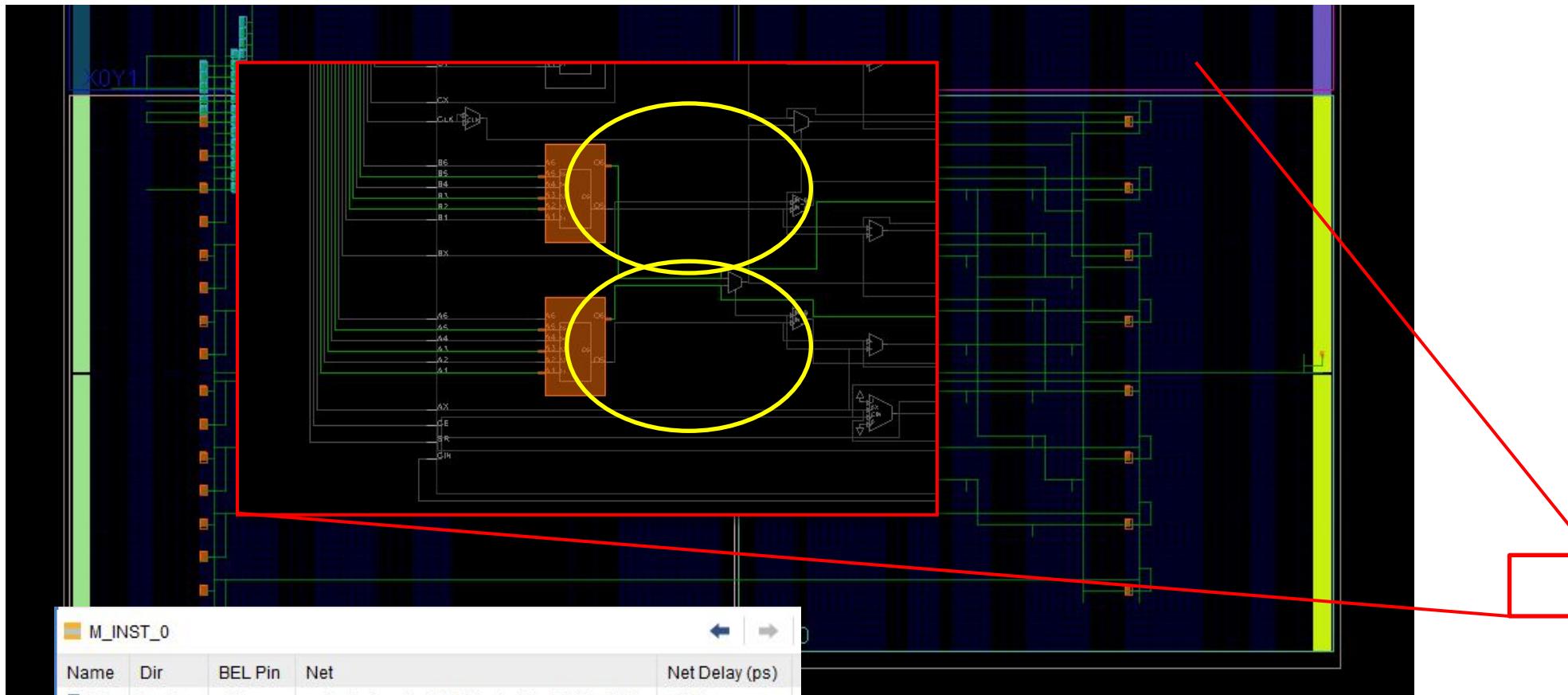
Ring Oscillator (RO) PUF



RO Place and Routing



RO Place and Routing



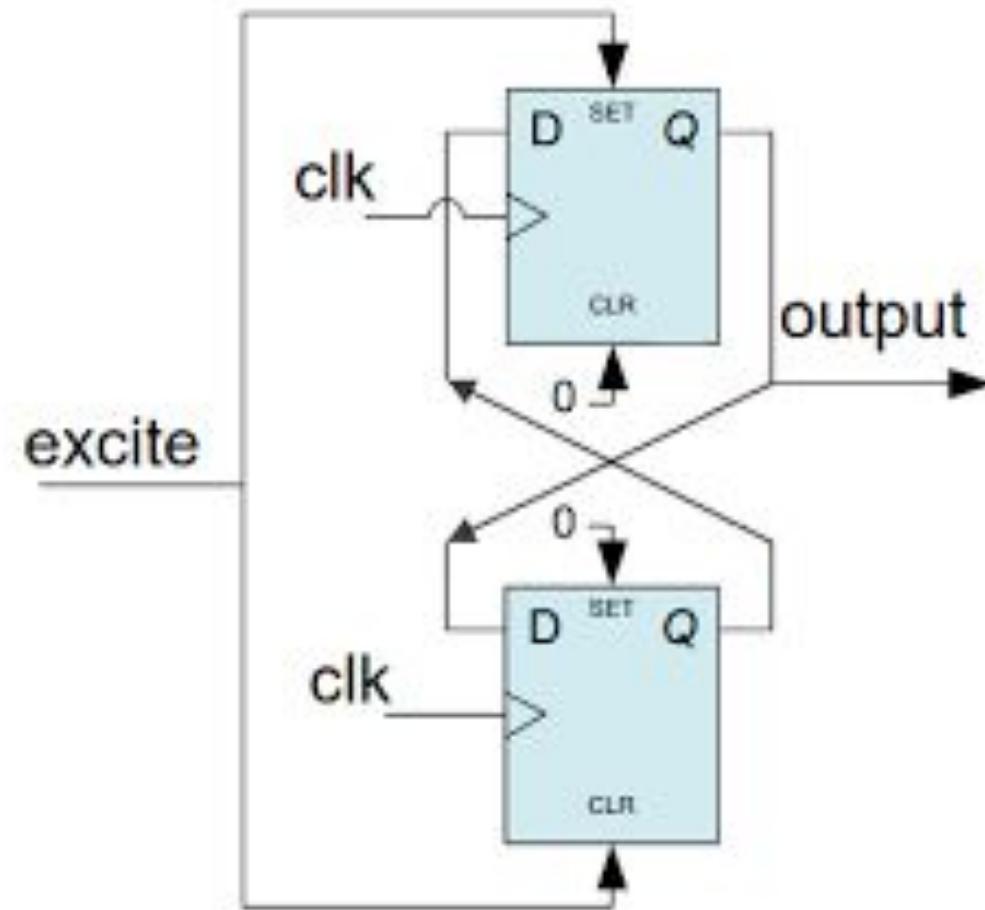
M_INST_0				
Name	Dir	BEL Pin	Net	Net Delay (ps)
I0	Input	A5	puf_design_i/puf8_0/inst/m1/m0/m0/d0	1959
I1	Input	A2	puf_design_i/puf8_0/inst/m1/m0/m0/m0/s	3840
I2	Input	A3	puf_design_i/puf8_0/inst/m1/m0/m0/m0/d1	1967
O	Output	O6	puf_design_i/puf8_0/inst/m1/m0/m0/M	0

Butterfly PUF

Cross Coupled DFFs.

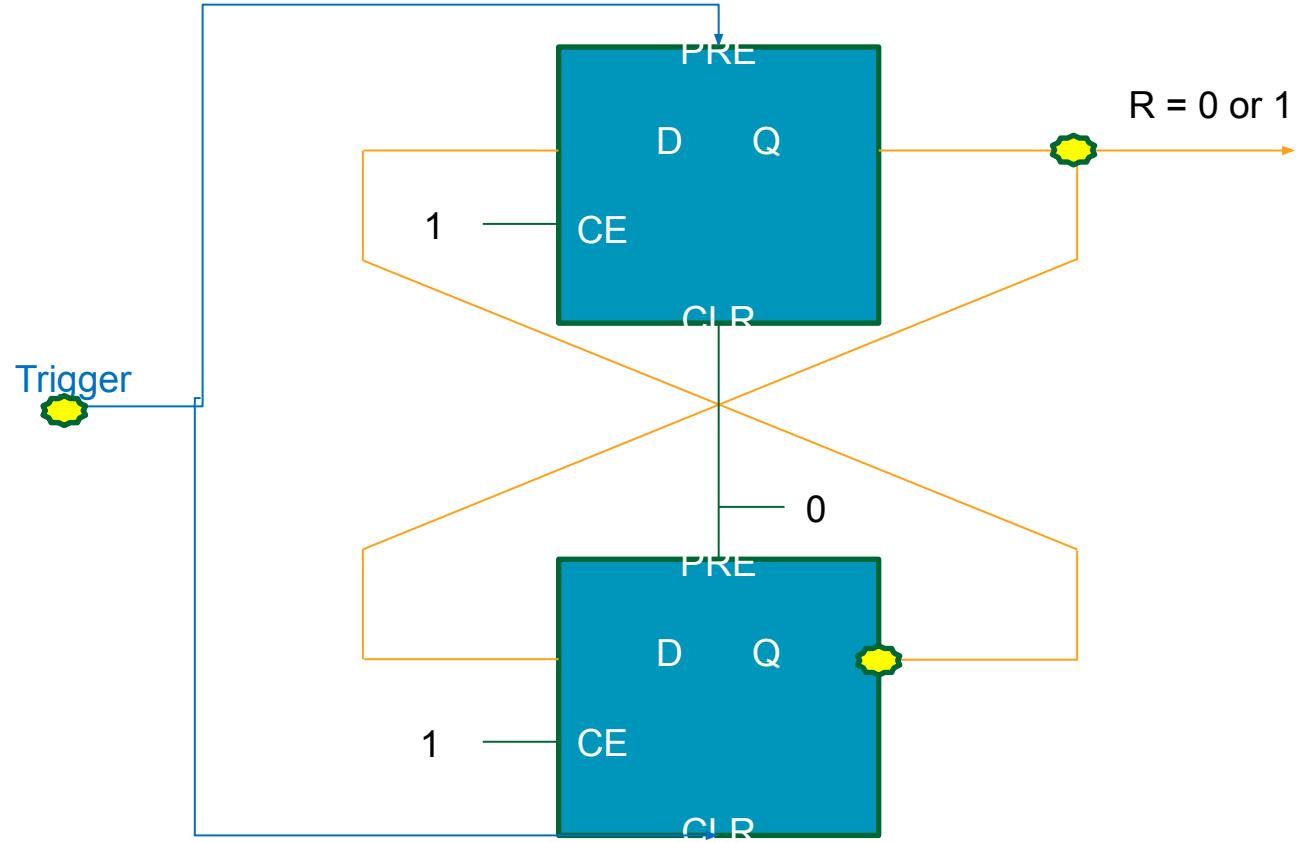
Physical ‘race condition’ easy to constrain compared to Arbiter.

Less machinery needed than RO PUF. (Less resources).



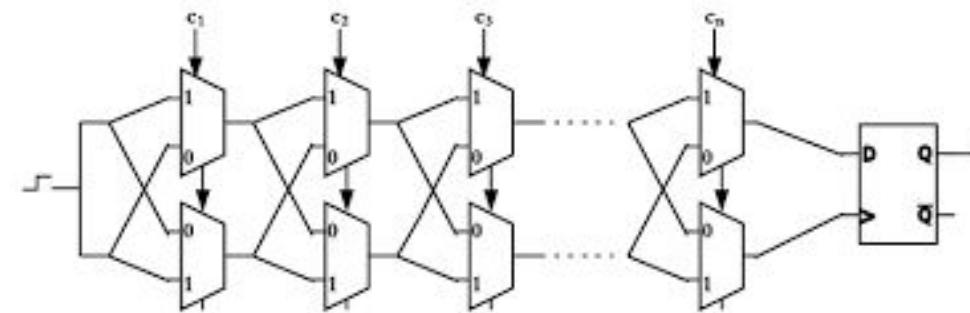
The Butterfly PUF

PRE	CLR	CE	D	Q
0	1	X	X	1
1	0	X	X	0
0	0	X	X	X
1	1	1	1	1
1	1	1	0	0
1	1	0	X	

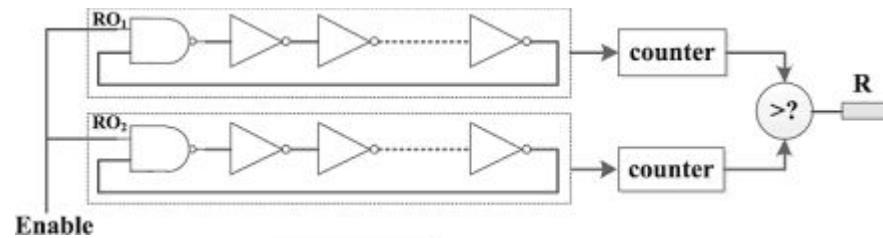


Three Primary PUF Types

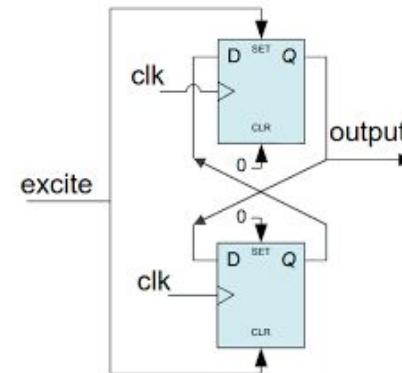
Arbiter: racing signals using MUXes
(path delay)



RO: counting signals
using path delay.



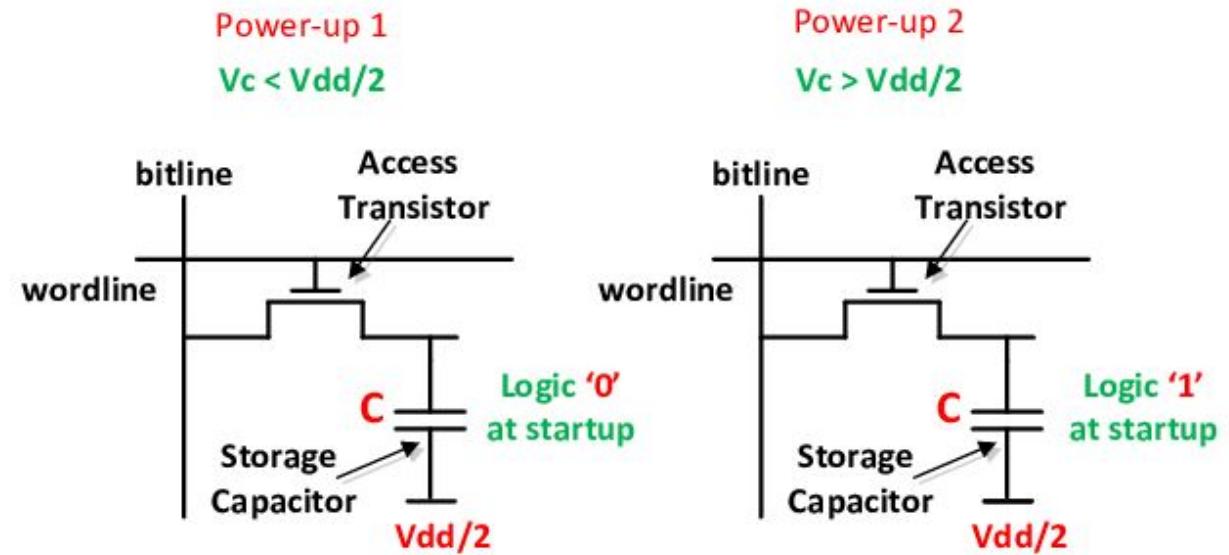
Butterfly: Activating
FFs using path delay.



DRAM PUF

Relies on biases in DRAM cell startup value.

Effectively isolating the leakage characteristics of cell capacitance.



What makes a good PUF?

1. Uniqueness!

- a. Ideally every challenge should elicit a single unique response.
- b. High Delta between two different challenge/response pairs (eg 50% hamming distance).

2. Difficult to model!

- a. resilient against ML / statistical methods.

3. Stable!

- a. stable over lifetime/age.
- b. stable against temp/environmental stimulus.

So what are PUFs actually useful for?

1. Authentication!

- a. If you know the unique challenge response (C/R) pairs, you can interrogate the PUF to ensure it is 'yours'.

2. Key Entropy!

- a. If you know C/R pairs, AND have a stable/reliable PUF, it can add entropy to key generation procedures.

3. Hiding/Obfuscating data!

- a. If you know your C/R pairs, a PUF can allow you to store information on chip without worrying about physically storing a key in silicon.

What Do PUFs Struggle With

1. Communication

- a. It is difficult to implement a PUF when the purpose is to secure a two way communication channel. More attack vectors to deal with (CPUF).
- b. Not impossible, but requires extra machinery (more attack vectors!).

2. Key Generation

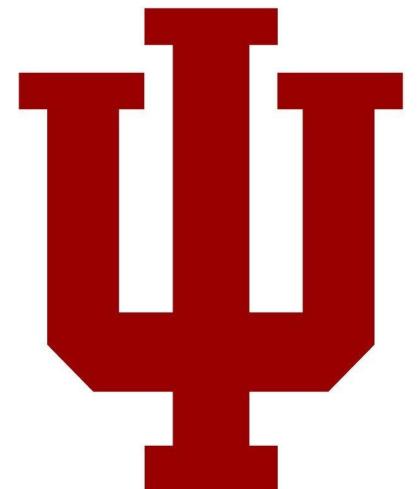
- a. Depending on a PUF for the entire key generation process can be risky due to reliability and stability concerns.

3. Difficult to Implement

- a. Implementation depends on isolating physical features of the target. Requires significant expertise in the target medium.

07 Truly Random Number Generators (TRNGs)

Engr 399/599: Hardware Security
Grant Skipper, PhD.
Indiana University



Adapted from: Mark Tehranipoor of University of Florida

Agenda

- Review PUFs
- Start & Finish TRNG
- Reminder Project Extension! -> This Friday! 2/21/24
Midnight!
- P2 Assigned Monday!

True Random Number Generator



Random Numbers in Cryptography

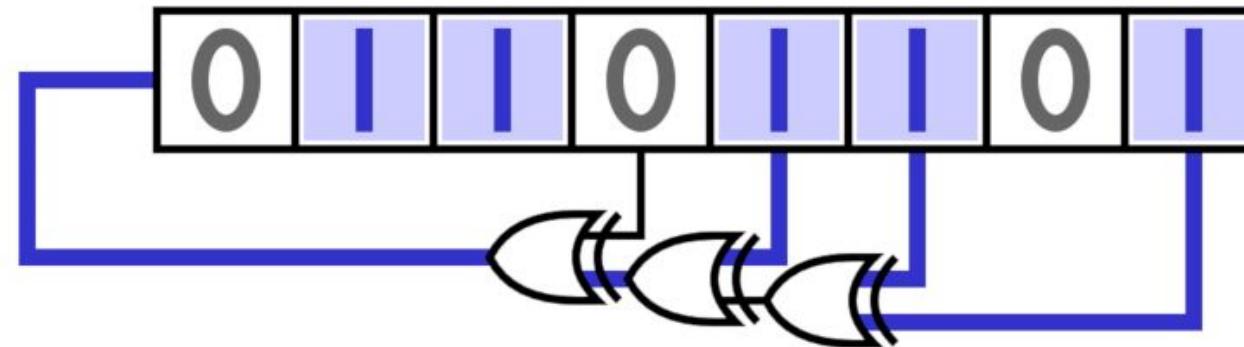
- The keystream in the one-time pad
 - The secret key in the DES encryption
 - The prime numbers p, q in the RSA encryption
 - Session keys
 - The private key in digital signature algorithm (DSA)
 - The initialization vectors (IVs) used in ciphers
-

Pseudo-random Number Generator

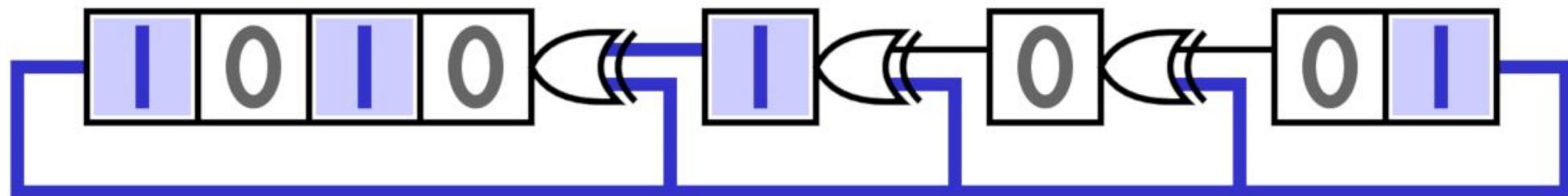
- **Pseudo-random number generator:**
 - A polynomial-time computable function $f(x)$ that expands a short random string x into a long string $f(x)$ that appears random
- **Not truly random in that:**
 - Deterministic algorithm
 - Dependent on initial values (seed)
- **Objectives**
 - Fast
 - Secure

Mersenne Twister (Pokemon)
LFSR (lazy cryptography)

Linear Feedback Shift Register

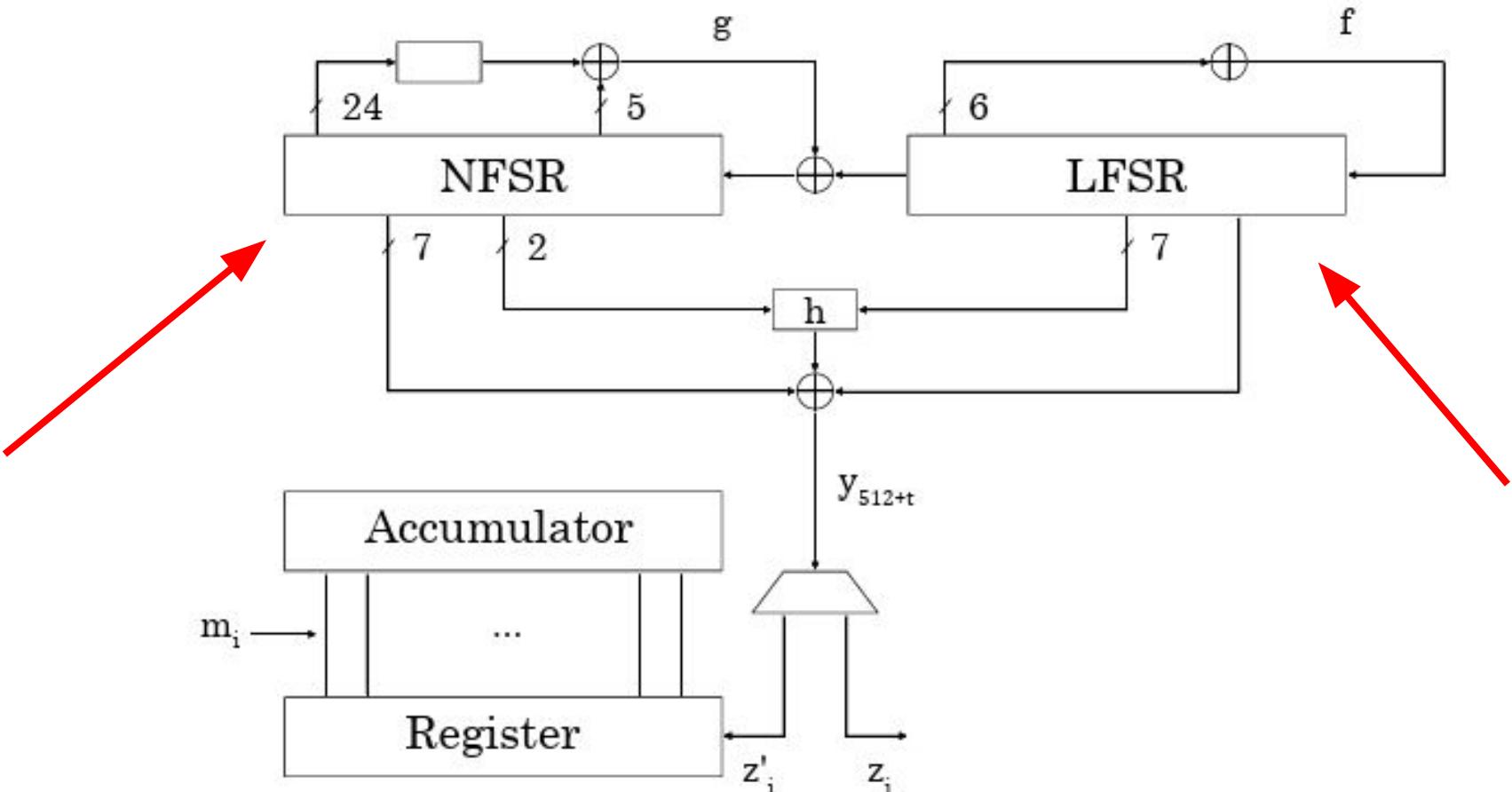


Fibonacci Configuration LFSR



Galois Configuration LFSR

LFSRs Still Used?

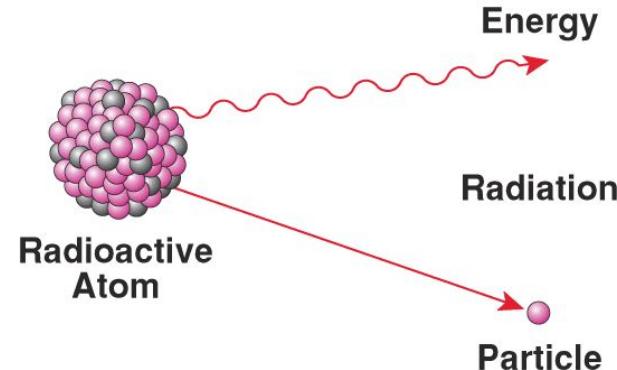
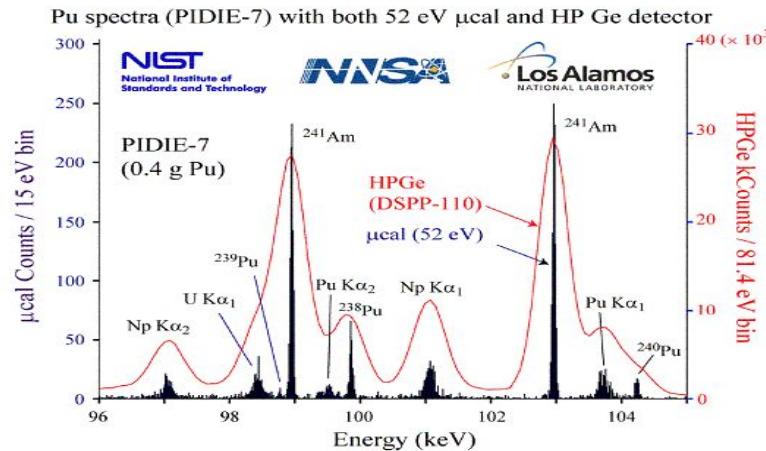


NIST LWC Grain128-AEAD (2021)

Sources

- The only truly random number sources are those related to physical phenomena such as **the rate of radioactive decay** of an element or the **thermal noise** of a semiconductor.

Lavarand!!!



- enomena. It is impossible to algorithmically generate truly random numbers.

Microcalorimeter (black) and high-purity germanium (red) spectra of a mixture of plutonium isotopes. Minimal thermal noise is achieved at 100 mK. High sensitivity is due to use of a superconducting quantum interference device.

Good TRNG Design

- **Entropy Source:**

- Randomness present in physical processes such as thermal and shot noise in circuits, brownian motion, or nuclear decay.

- **Harvesting Mechanism:**

- The mechanism that does not disturb the physical process but collects as much entropy as possible.

- **Post-Processing (optional):**

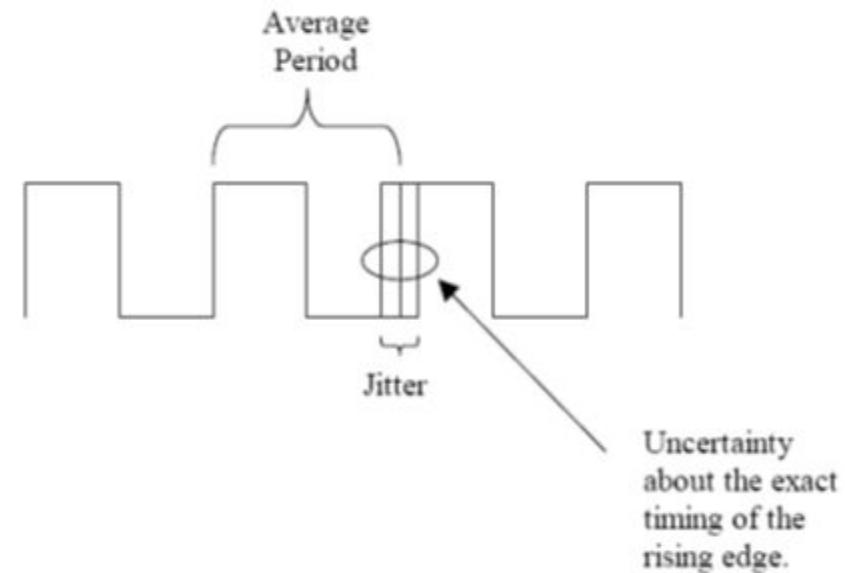
- Applied to mask imperfections in entropy sources or harvesting mechanism or to provide tolerance in the presence of environmental changes and tampering.

Set of Requirements

- ❑ The Design Should be purely digital
 - ❑ The harvesting mechanism should be simple.
 - The unpredictability of the TRNG should not be based on the complexity of the harvesting mechanism, but only on the unpredictability of the entropy source.
 - ❑ No correction circuits are allowed
 - ❑ Compact and efficient design (high throughput per area and energy spent).
 - ❑ The design should be sufficiently simple to allow rigorous analysis.
-

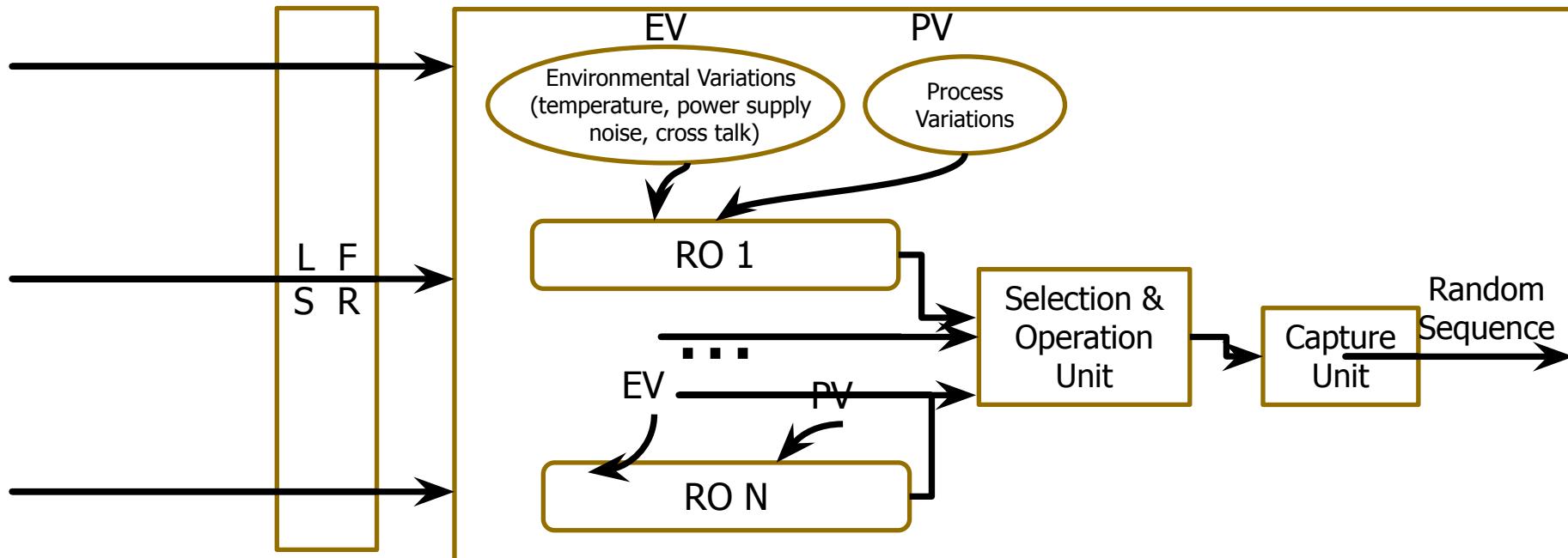
Method : Clock Jitter

- Jitter is variations in the significant instants of a clock
 - Jitter is nondeterministic (random)
-
- **Sources of Jitter:**
 - Semiconductor noise
 - Cross-talk
 - Power supply variations
 - Electromagnetic fields



TRNG Structure

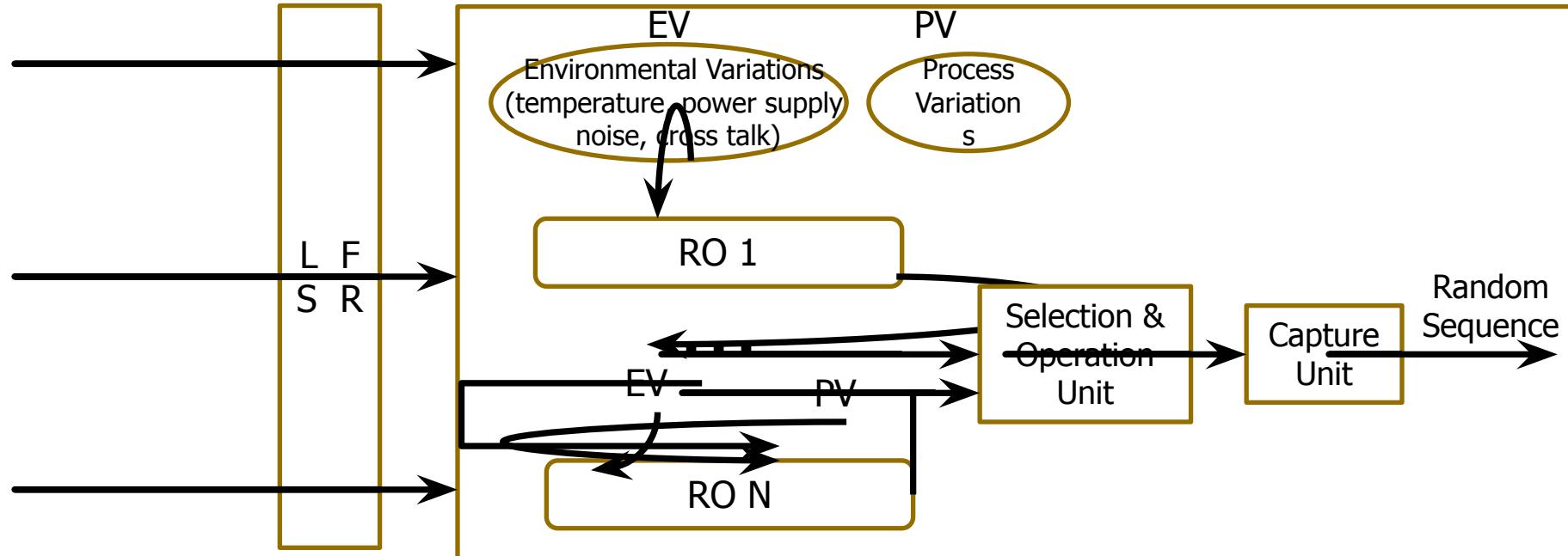
- ❑ **LFSR:** Generate random patterns, causing random switching noise



TRNG Structure

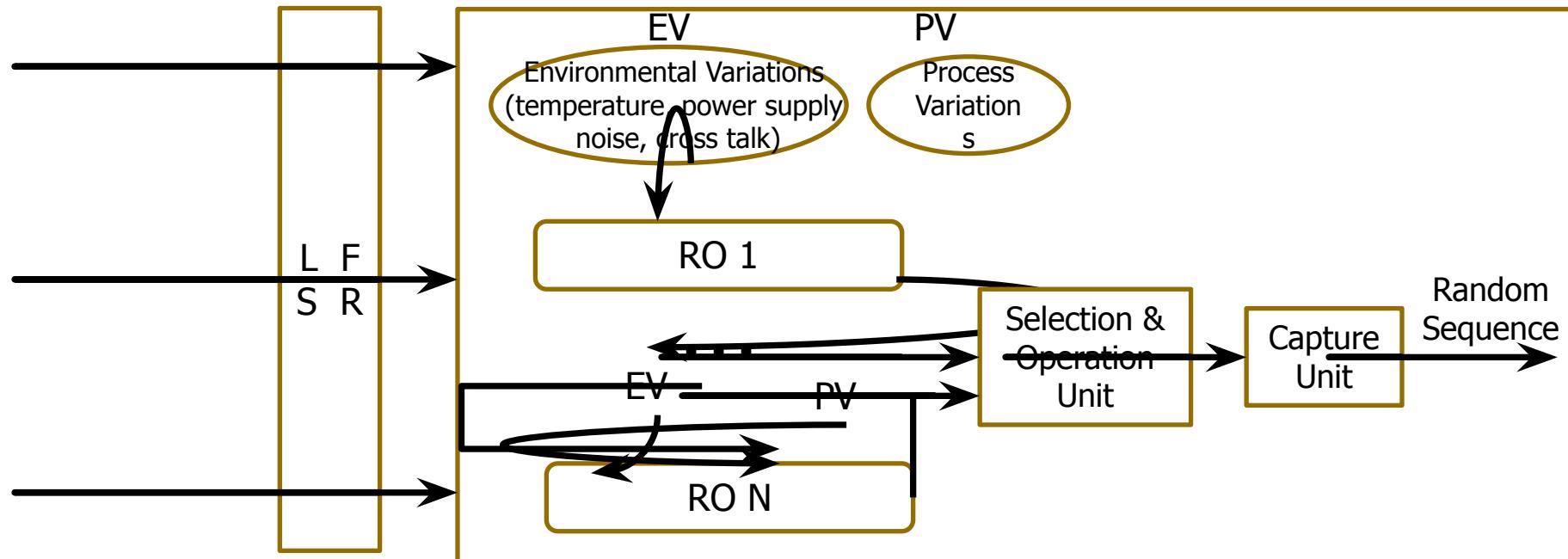
□ Ring Oscillators

- Process variations & environmental variations
- Random phase jitter



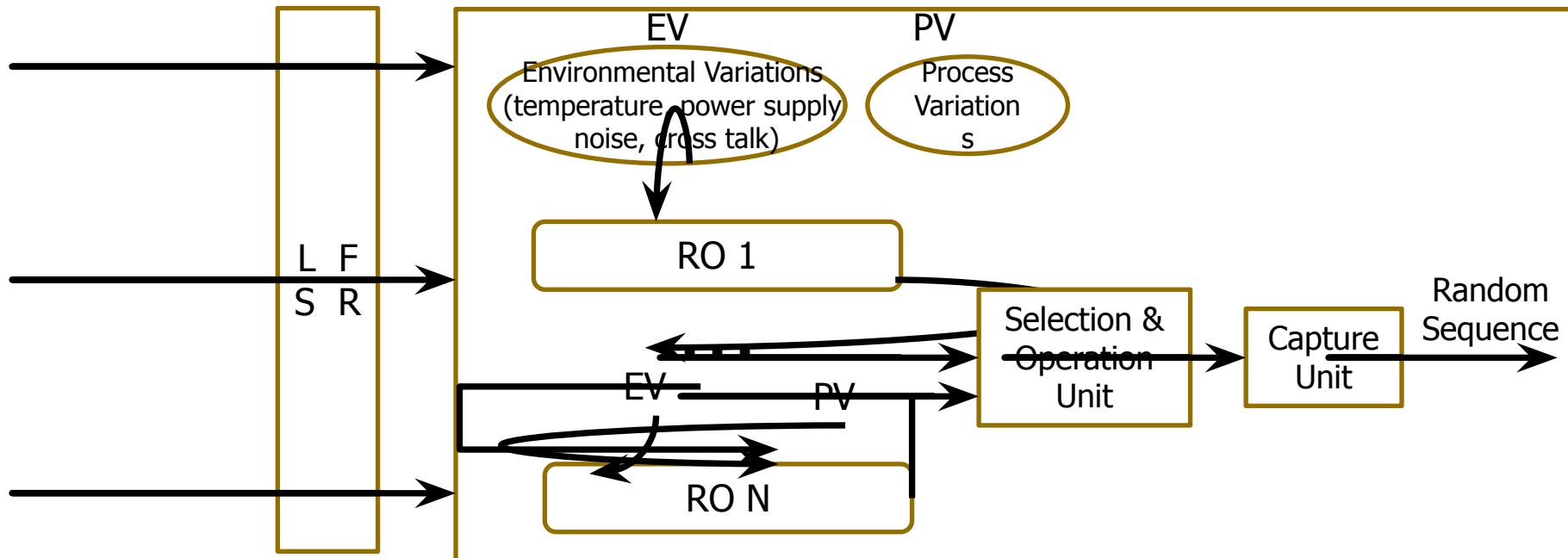
TRNG Structure

- **Selection & Operation Unit:** The random phase of ring oscillators could be translated into digital values by this unit, such as XOR operation

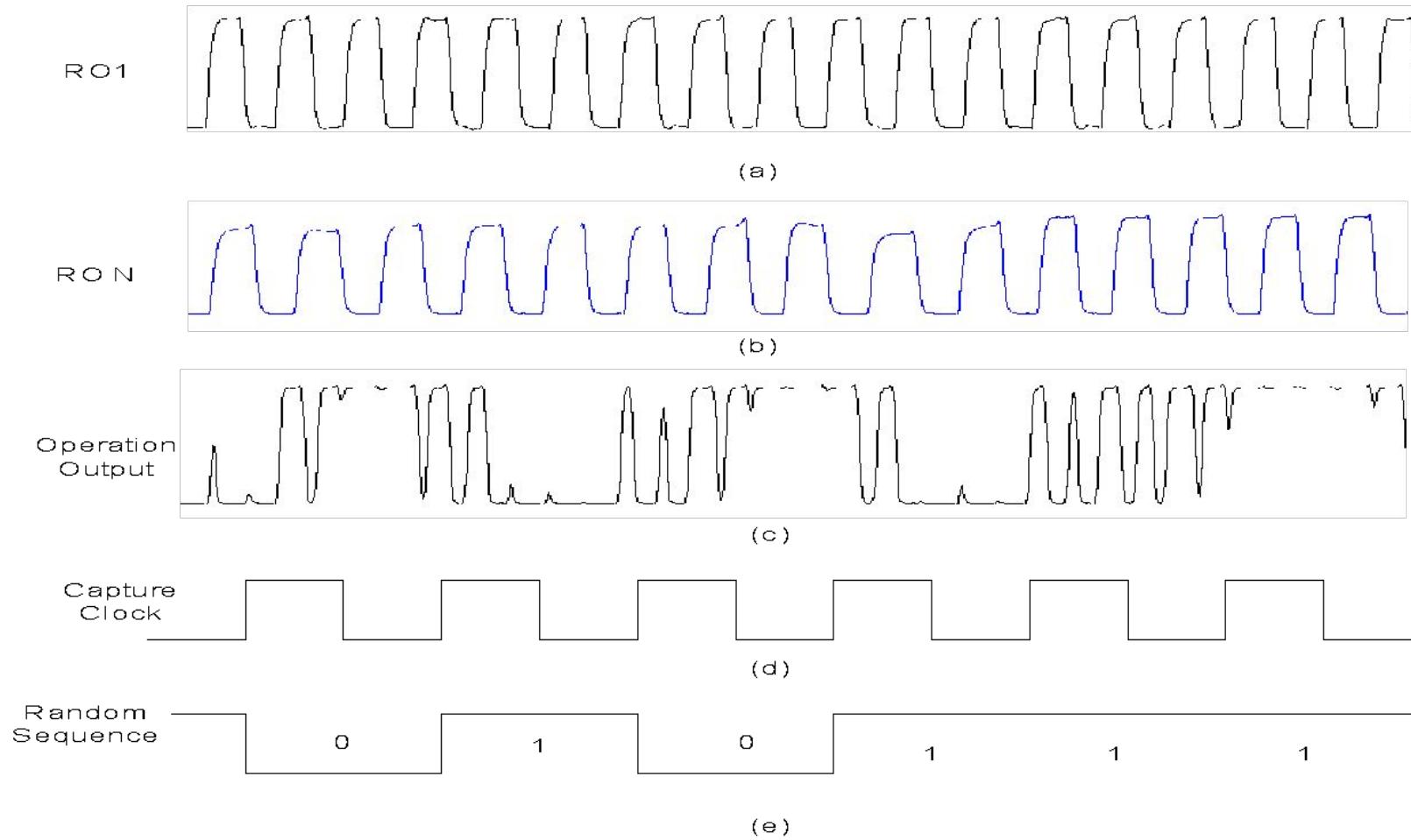


TRNG Structure

- **Capture Unit:** Make sure the digital value is sampled with the frequency of the required true random number.



TRNG Output



References

- [1]Suh, G.E., Devadas, S.: Physical unclonable functions for device authentication and secret key generation. In: Design Automation Conference, pp. 9{14. ACM Press, New York, NY, USA (2007)
- [2]Gassend, B., Lim, D., Clarke, D., van Dijk, M., Devadas, S.: Identification and au-thentication of integrated circuits: Research articles. *Concurr. Comput. : Pract. Exper.* 16(11), 1077-1098.
- [3]Gassend, B., Clarke, D., van Dijk, M., Devadas, S.: Controlled physical random functions. In: ACSAC '02: Proceedings of the 18th Annual Computer Security Applications Conference, p. 149. IEEE Computer Society, Washington, DC, USA (2002)
- [4]B. Gassend, D. Clarke, M. van Dijk, and S. Devadas. Silicon physical random functions. In Proceedings of the Computer and Communication Security Conference , November 2002.
- [5] Dinesh Ganta, Vignesh Vivekraja, Kanu Priya and Leyla Nazhandali, “A Highly Stable Leakage-Based Silicon Physical Unclonable Functions”

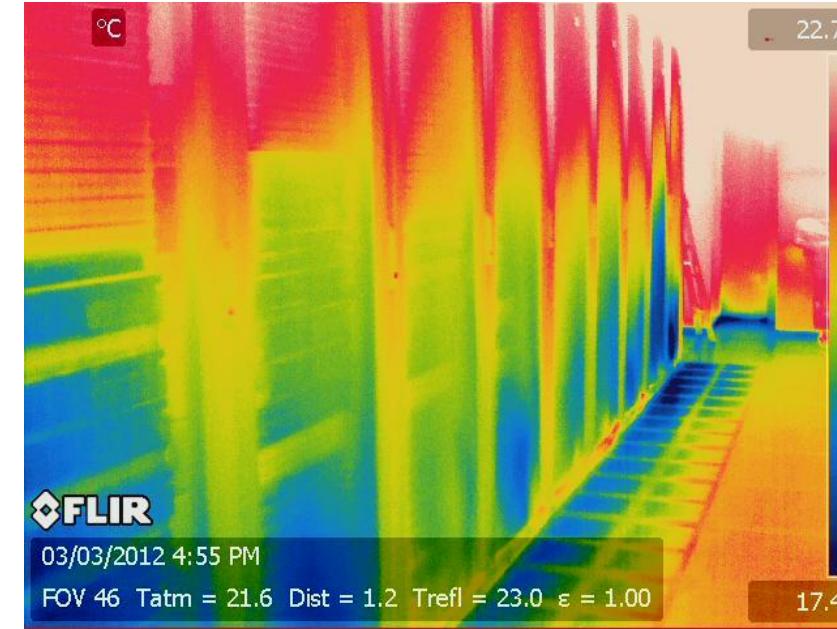
References

- [6] A. Maiti and P. Schaumont, “Improved ring oscillator puf: An fpga-friendly secure primitive,” J. Cryptology, vol. 24, no. 2, pp. 375–397.,2011.
- [7] B. Sunar, W. J. Martin, D. R. Stinson. A Provably Secure True Random Number Generator with Built-in Tolerance to Active Attacks. IEEE Transactions on Computers, vol 58, no 1, pages 109-119, January 2007.

More on PUF Applications

Motivation: Securely Sensing without Key

- Physical features should be closely monitored
 - The temperature of data center, Facebook, Google, Amazon, etc



Motivation: Securely Sensing without Key

- Physical features should be closely monitored
 - The temperature of data center, Facebook, Google, Amazon, etc
- Some physical features might be meaningful for security/privacy
 - Relative location between bank cards and automated teller machines (ATMs), card (not) present withdrawal

Motivation: Securely Sensing without Key

- Physical features should be closely monitored
 - The temperature of data center, Facebook, Google, Amazon, etc
- Some physical features might be meaningful for security/privacy
 - Relative location between bank cards and automated teller machines (ATMs), card (not) present withdrawal



Motivation: Securely Sensing without Key

- Physical features should be closely monitored
 - The temperature of data center, Facebook, Google, Amazon, etc
 - Some physical features might be meaningful for security/privacy
 - Relative location between bank cards and automated teller machines (ATMs), card (not) present withdrawal
 - Some physical features are hard to sense but favoring many applications
 - Digital rights management, physically/irreversibly canceling membership?
 - Secure sensors are needed:
 - Operation safety, secrecy of sensitive data
-

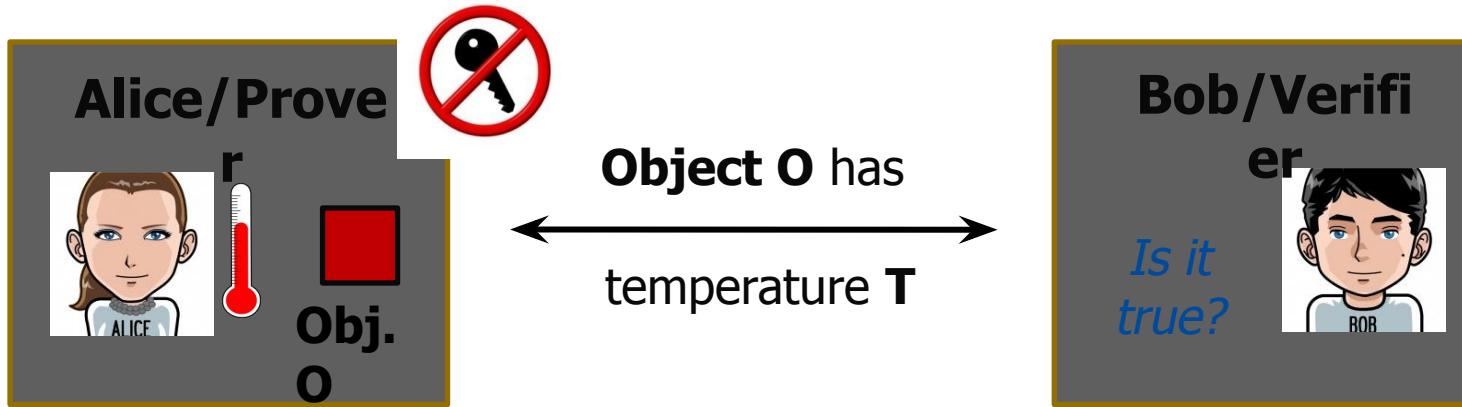
Key Is a Target

- Modern security protocols are commonly based on secret keys.
- A robust key enhances the robustness of security systems, but also announces itself as an interested target for attackers. [1]



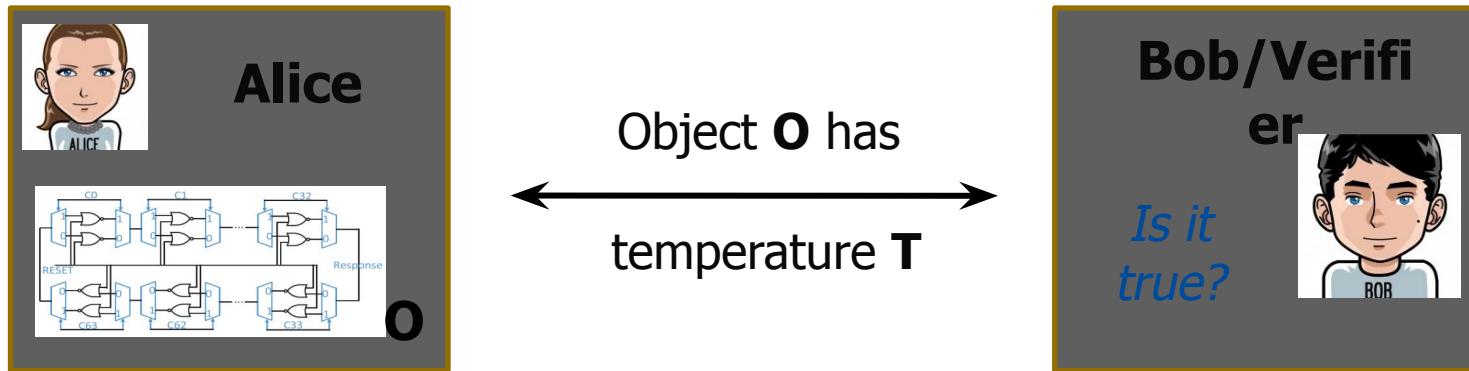
[1] Rivest, Ronald L. Illegitimi non carborundum. CRYPTO (2011).

Virtual Proofs of Temperature (Or: Keyless Temperature Sensors)



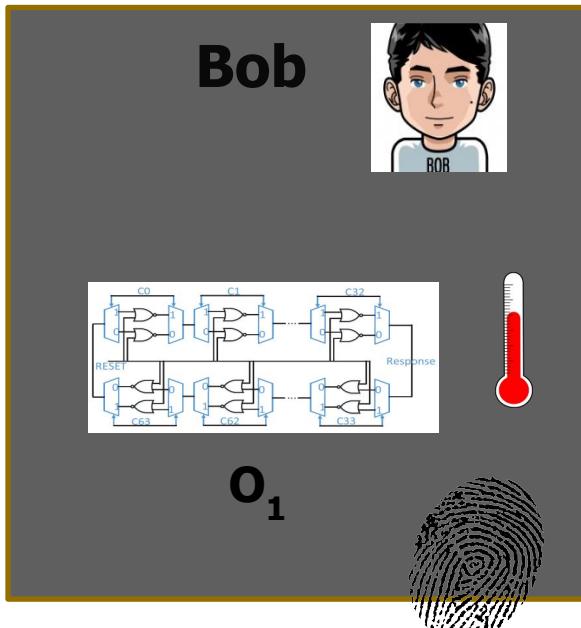
- Alice wants to prove to Bob that
 - a certain **object O**
 - is at temperature **T**
 - at the time of the execution of the VP
- **Classical approach:** Secure sensors with key known to Bob...
 - But, of course, we want no keys...

VPs of Temperature via Noise Sensitivity



- **Key Idea:** Use a *temperature-sensitive Strong PUF* as Object O
 - ie., an integrated circuit (IC) / PUF whose output **intentionally** depends upon the ambient temperature T
 - Output of the IC/PUF to Bob's random challenges proves temperature
-

VP of Temperature $T_i \in \{T_1, \dots, T_k\}$



Private set-up phase:

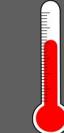
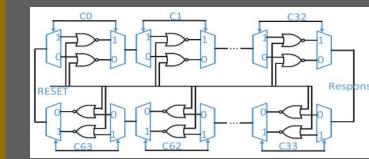
- For each temperature $T_i \in \{T_1, \dots, T_k\}$:
 - Bob puts the objects O_1 at temperature T_i
Chooses n random challenges C_1^i, \dots, C_n^i
Measures the n resulting responses: r_1^i, \dots, r_n^i
 - Bob creates&stores private list $\text{List}(T_i) = (C_1^i, r_1^i), \dots, (C_n^i, r_n^i)$

VP of Temperature $T_i \in \{T_1, \dots, T_k\}$



Alice

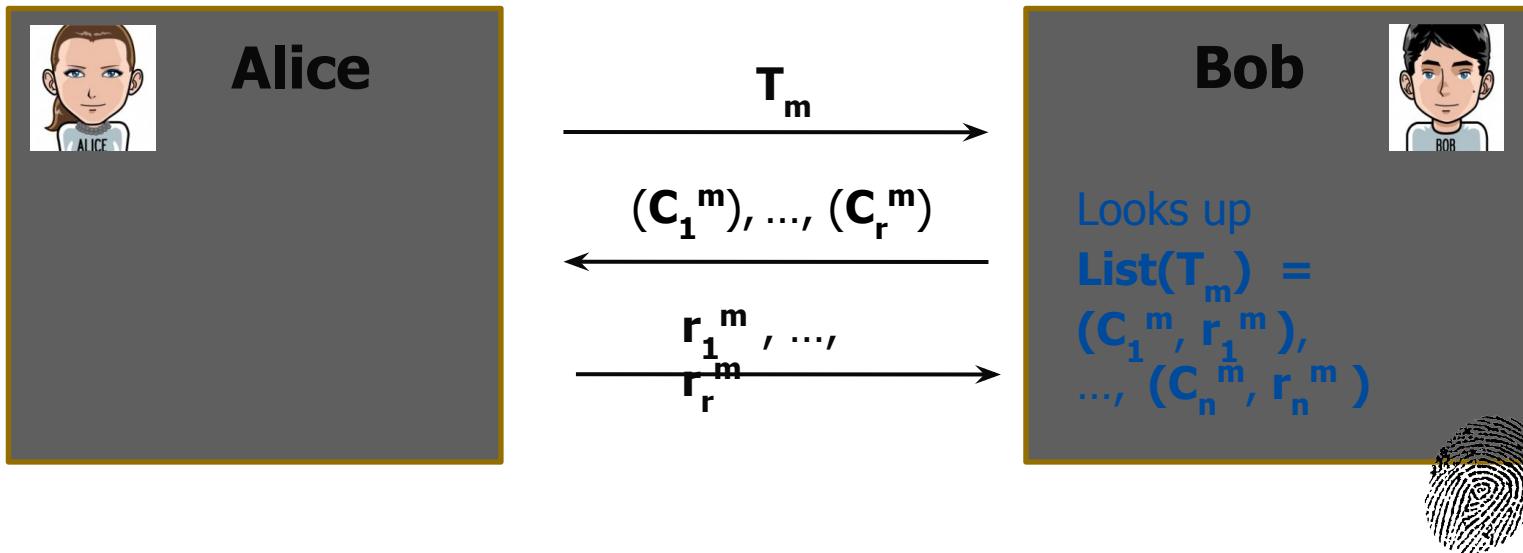
Bob



- Set-up phase closes, and the objects are transferred to **Alice**
- Some time may pass...
(time for Alice to attack the system...)
- **VP starts!!**



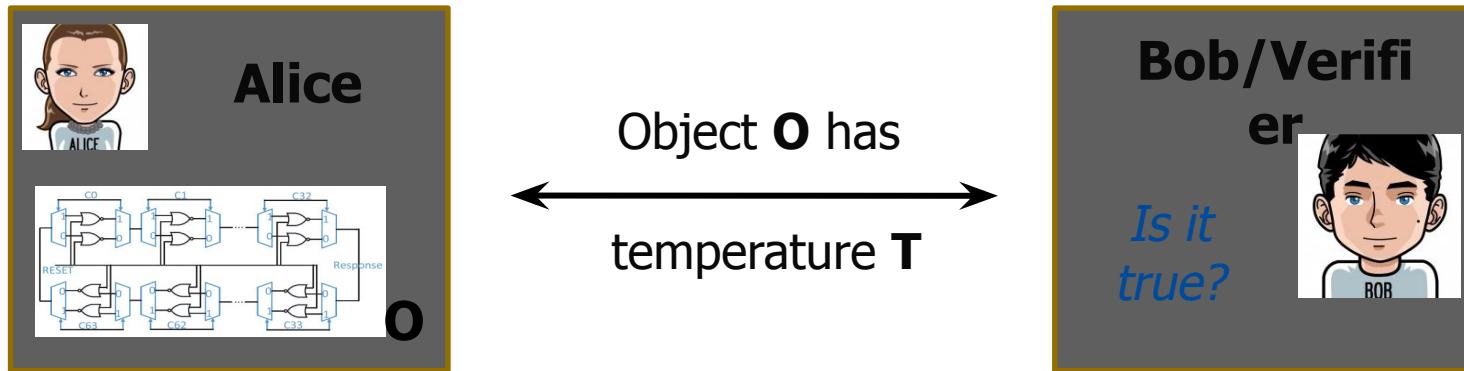
VP of Temperature $T_i \in \{T_1, \dots, T_k\}$



- If pre-recorded values I_j^m in the $\text{List}(T_m)$ match the values that **Alice** sent, then **Bob** will **accept** the VP



VPs of Temperature via Noise Sensitivity



- Leads to „keyfree temperature sensors“
 - Practical outcome of our new theoretical concept!!!
- Temp.-dependent behavior in PUFs usually **unwanted**
 - Turning known weakness of PUF into strength!
- **Again**, all standard properties of PUFs needed
 - Unclonability, no modeling, many challenge-response pairs