05 Physically Unclonable Functions (PUFs)

^ not this

Engr 399/599: Hardware Security

Grant Skipper, PhD.
Indiana University

Adapted from: Mark Tehranipoor of University of Florida

Course Website

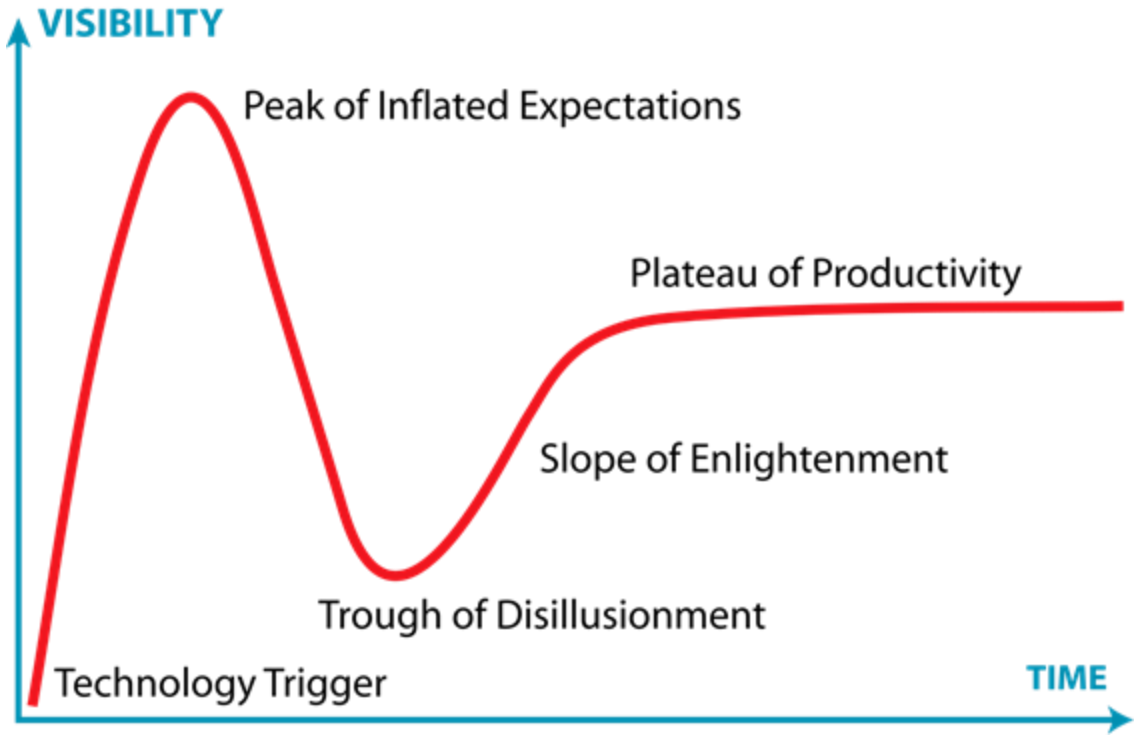
engr599.github.io

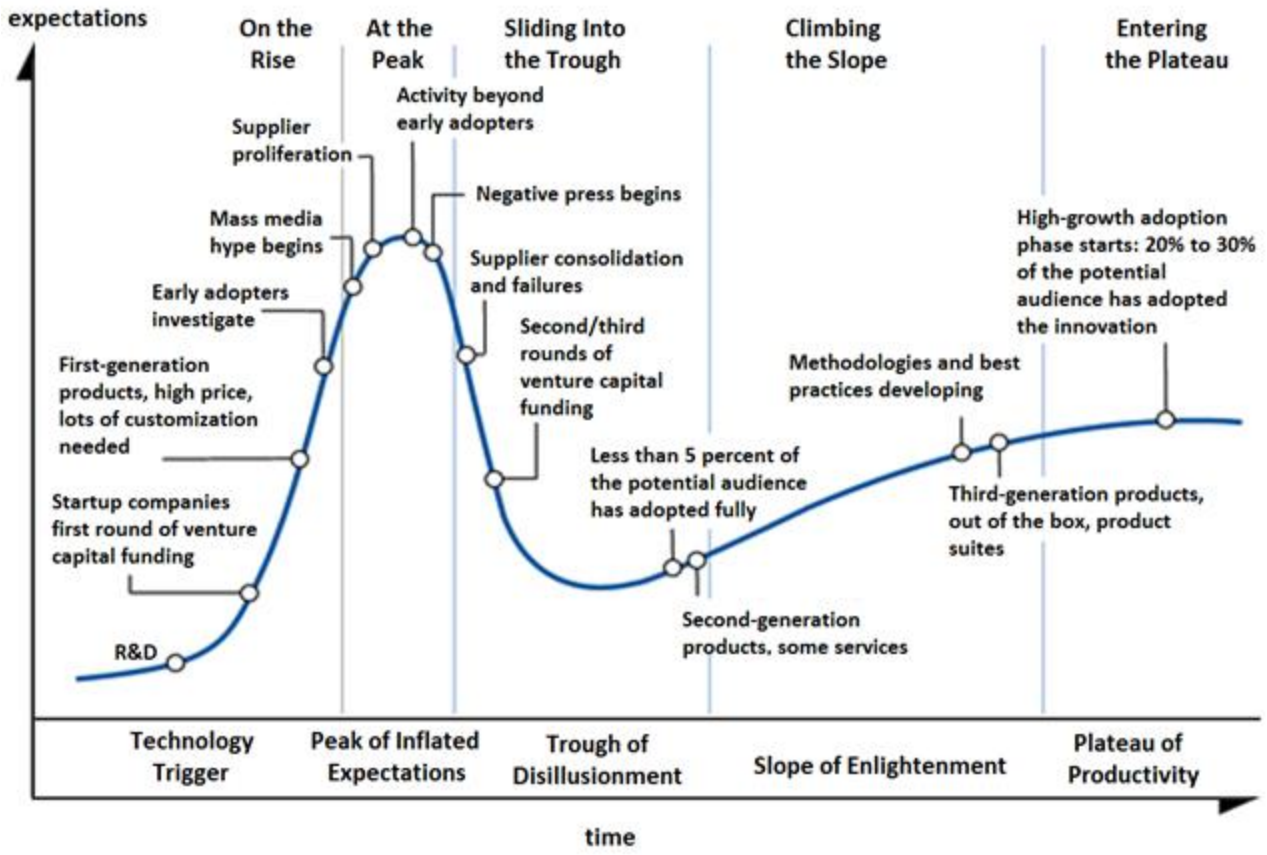
Write that down!

Agenda

- Review VLSI
- PUFs!
- Project!

Gartner Hype Cycle





Side Quest: Hardware Backdoor in Keycards!

https://thehackernews.com/2024/08/hardware-backdoor-discovered-in-rfid.html - Credit, Quarkslab.

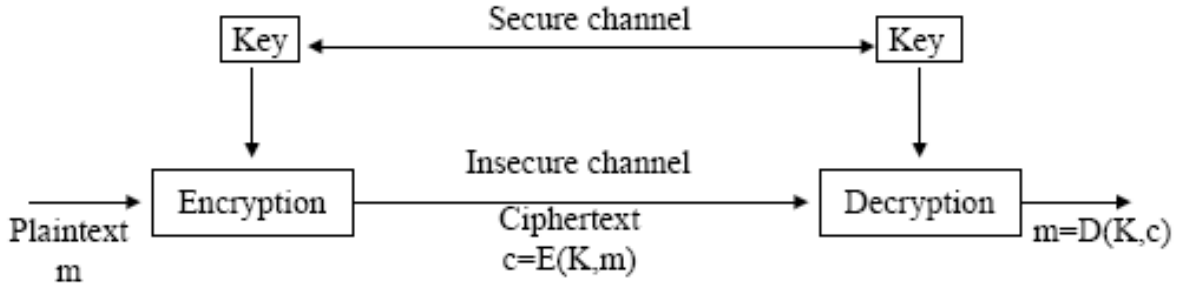
"Apparently, all FM11RF08S implement a backdoor authentication command with a unique key for the entire production…."

What is FM11RF08S?

RFID chips so popular to attack the vendor is breaking it for us!

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

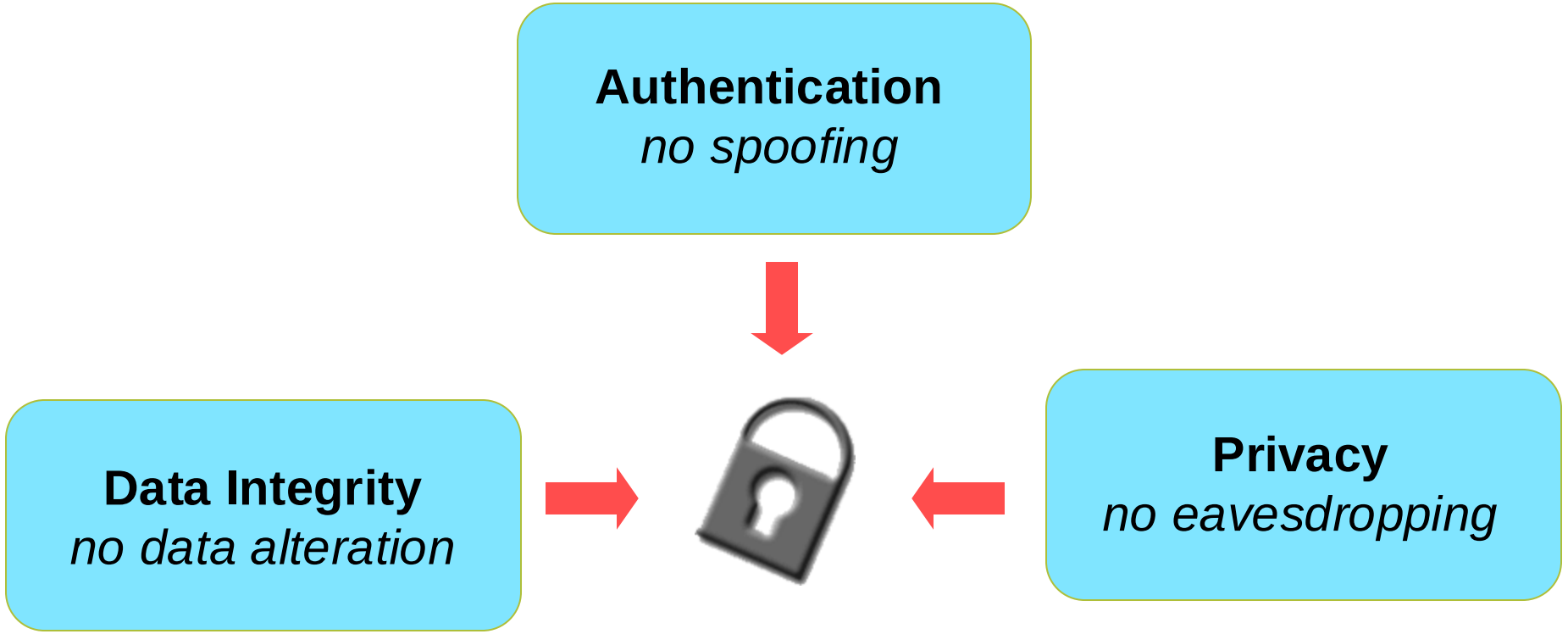
Digital Storage

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

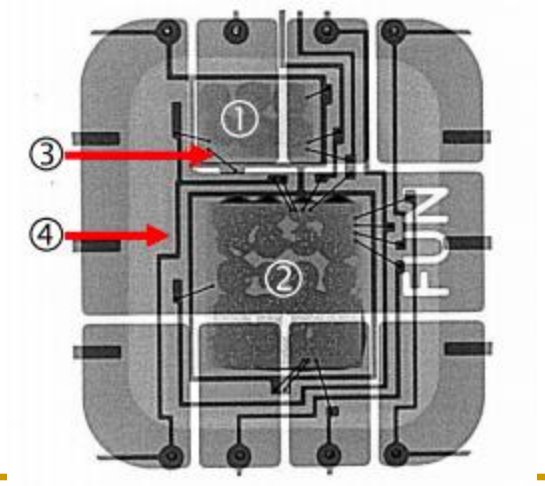
Software-only protection is not enough. Non-volatile memory technologies are vulnerable to invasive attack as secrets always exist in digital form

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.

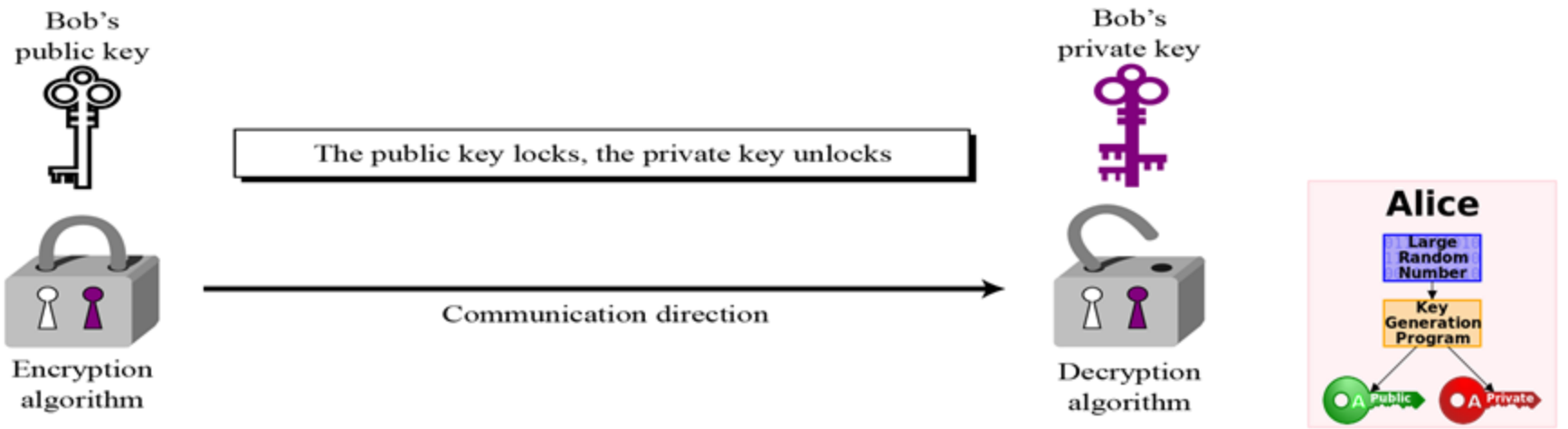
https://arstechnica.com/tech-policy/2015/10/how-a-criminal-ring-defeated-the-secure-chip-and-pin-credit-cards/

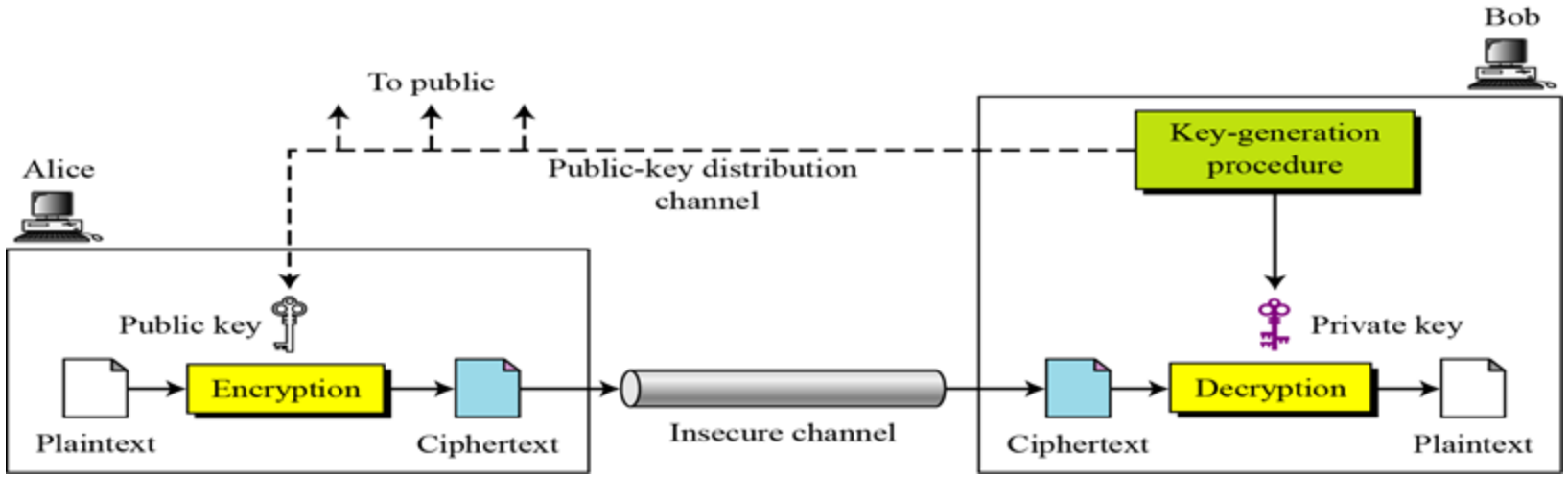


Basic Terminologies

- 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

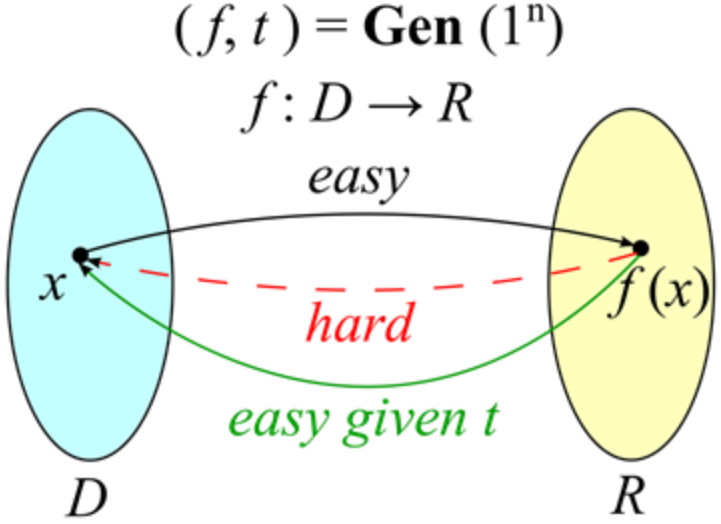
Challenges of algorithmic (mathematical) one-way functions

- **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

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!



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

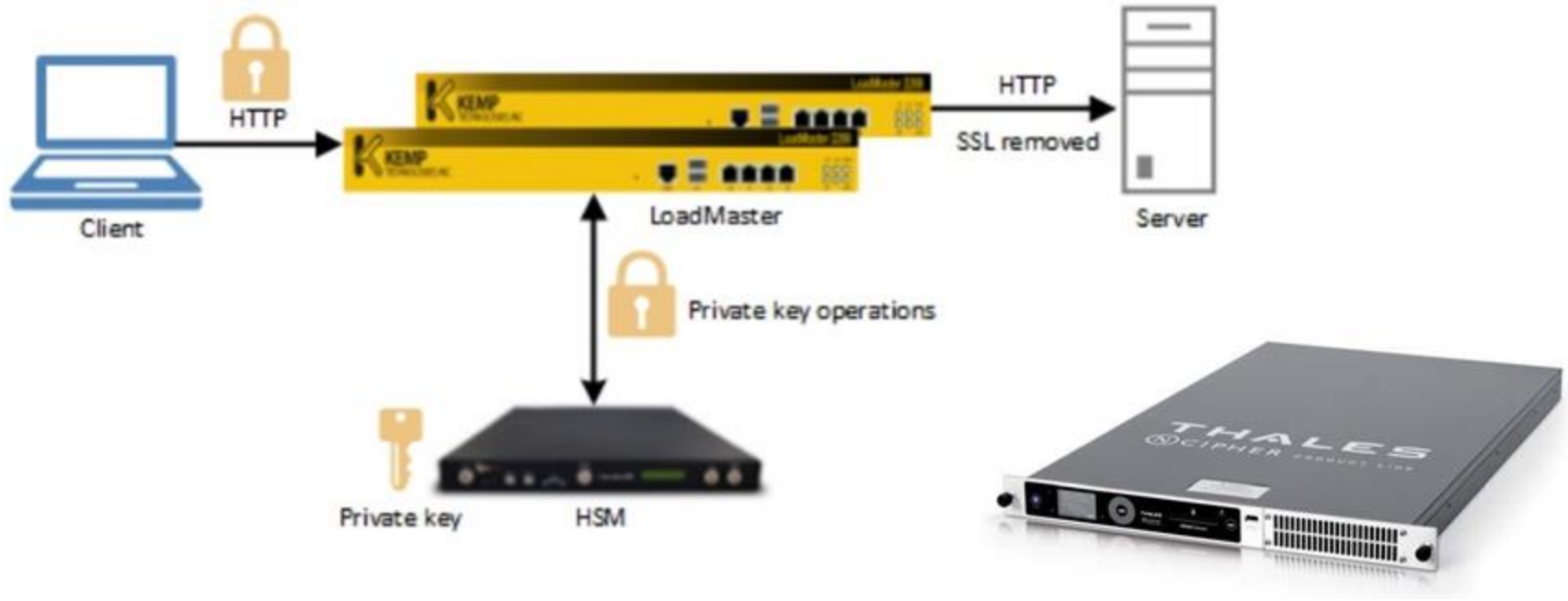
- Tens of sensors, resistance, temperature, voltage, etc.

- Continually battery-powered

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

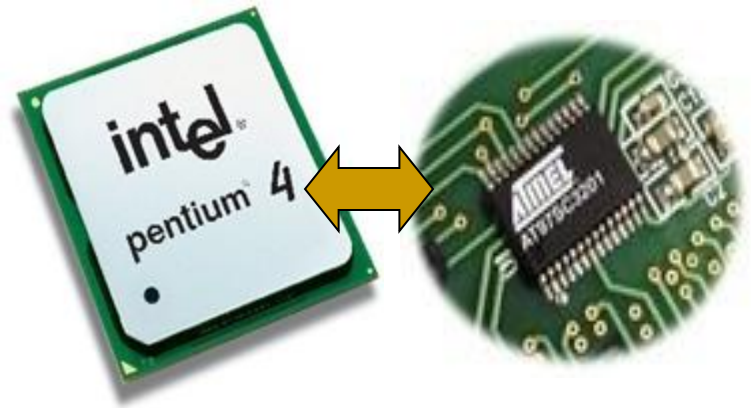
HSM

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



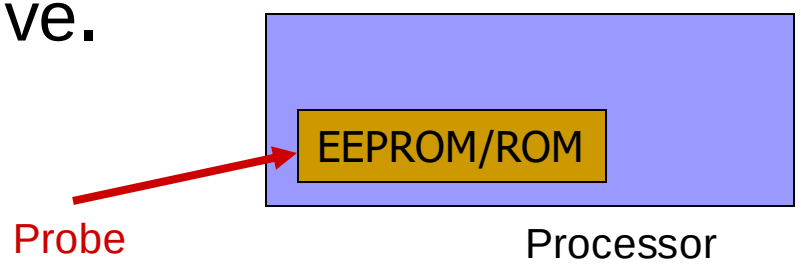
A separate chip (TPM) for security functions

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.



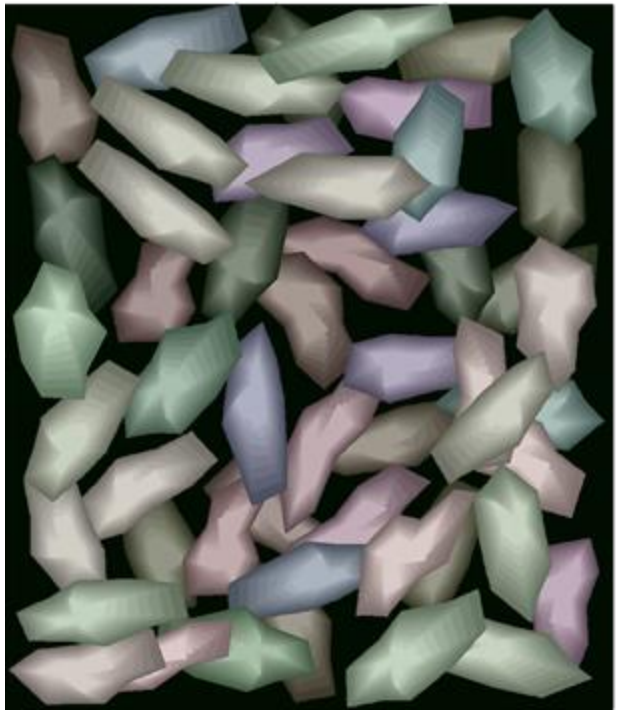
- Adversaries can physically extract secret keys from EEPROM while processor is off
- Trusted party must embed and test secret keys in a secure location
- EEPROM adds additional complexity to manufacturing

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.

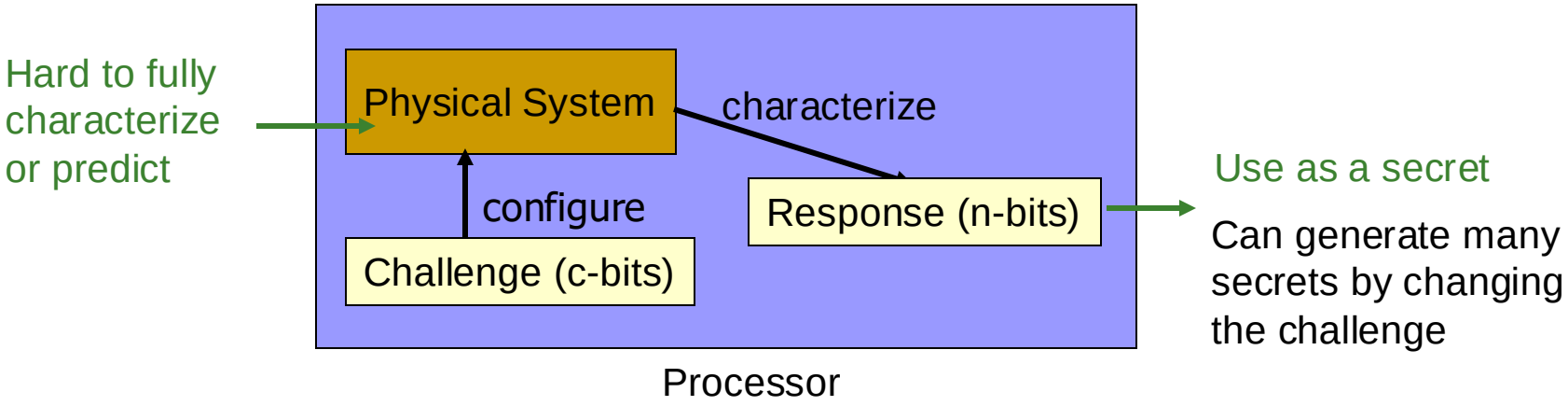


"In information theory, the entropy of a random variable quantifies the average level of uncertainty or information associated with the variable's potential states or possible outcomes."
- Wikipedia

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?

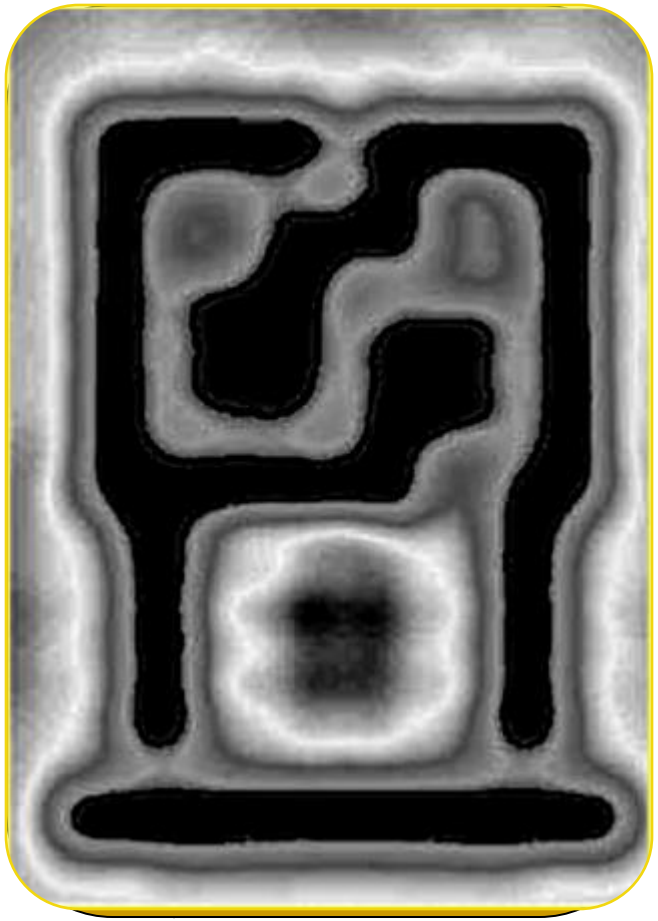
Definition

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

WYSINWYG

Sub-Wavelength WYSINWYG



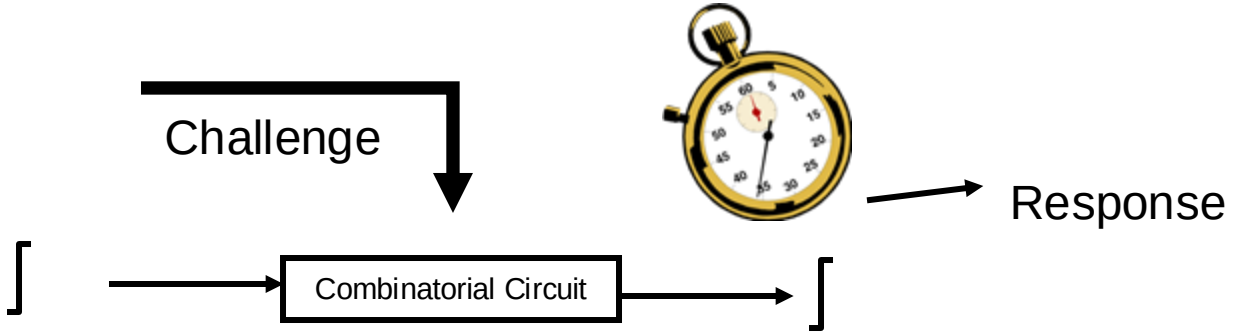
What You See Is Not What You Get

Process variations

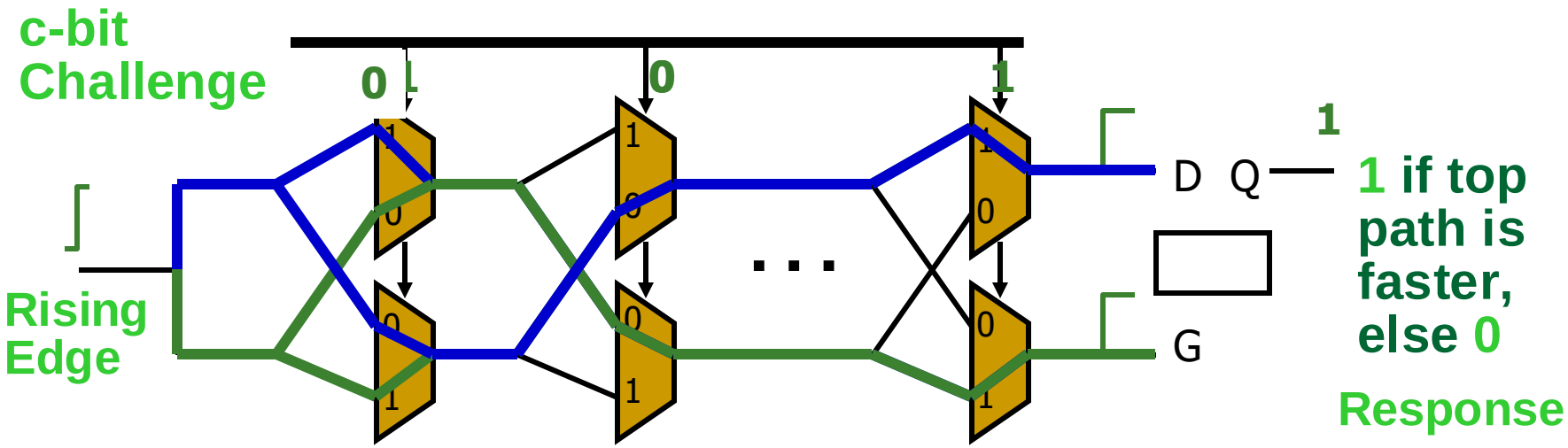
No two transistors have the same parameters

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.



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.

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.

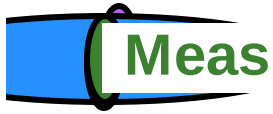
Experiments

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

100 bits of response

Distance between Chip X and Y responses = 24 bits

At 70C measurement noise for chip X = 2



Measurement noise for Chip X = 0.5

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 X = 2

Measurement noise for Chip X = 0.5

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 X = 2

Measurement noise for Chip X = 0.5

Model-building appears hard even for simple circuits

“Best” model for Chip X has error = 10

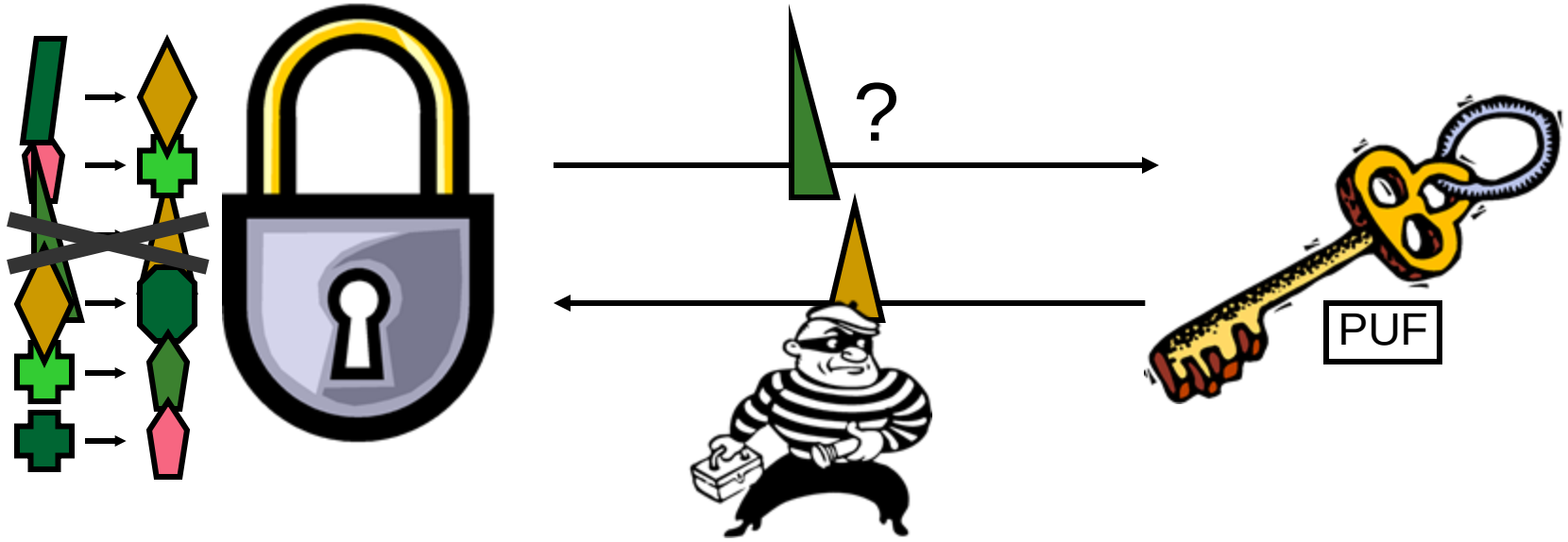
Physical Attacks

- 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.**



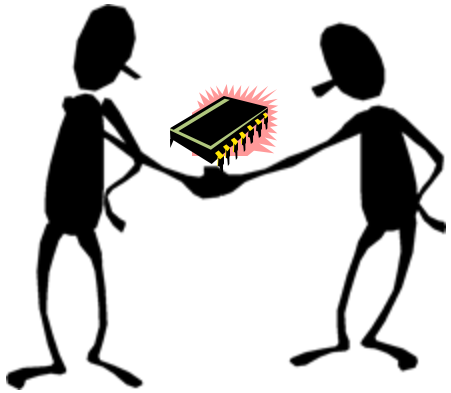
Applications

- **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.



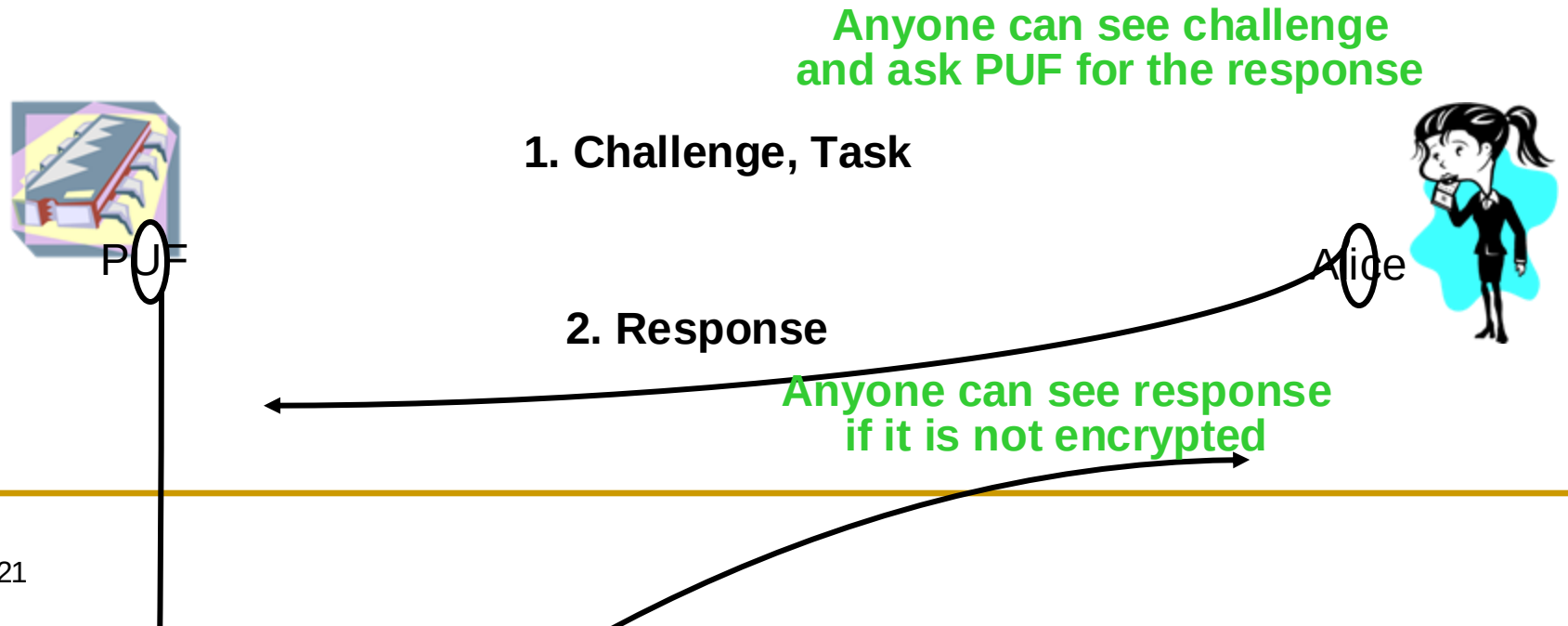
We can enable the above applications by trusting only a single-chip processor that contains a silicon PUF.

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

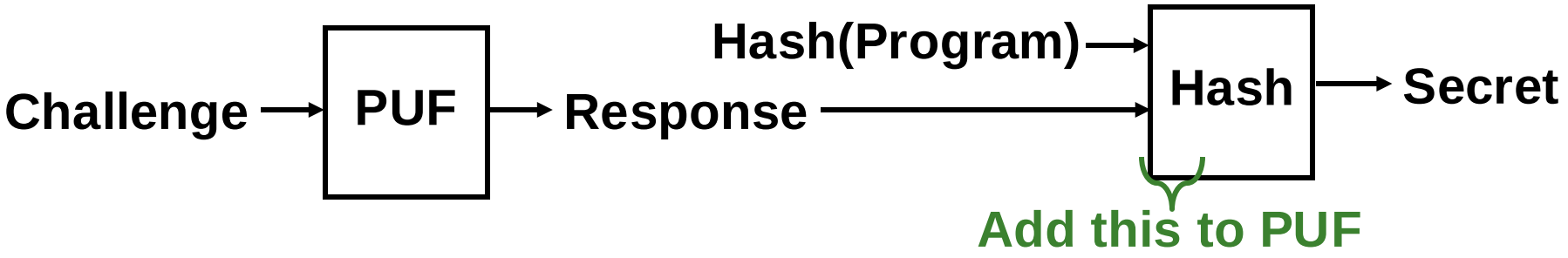


Restricting Access to the PUF

- 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.

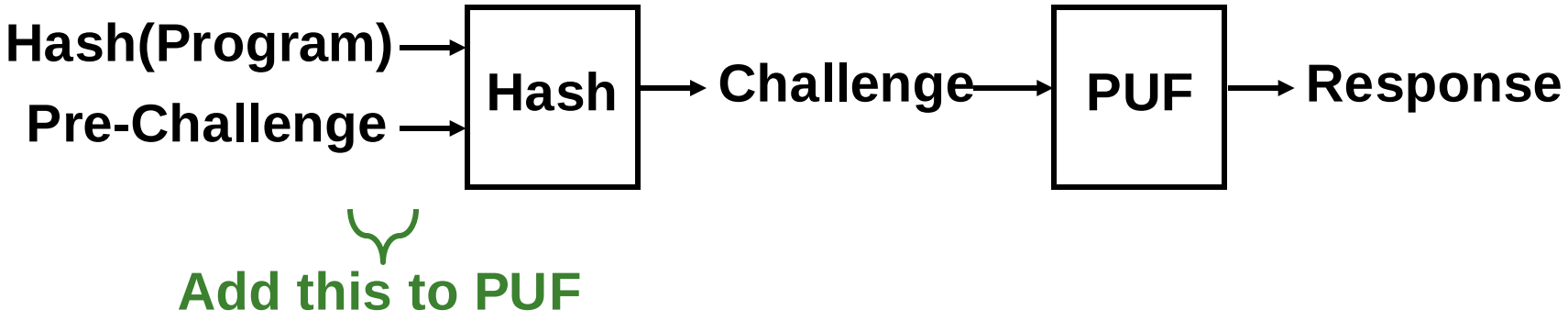
- But Alice can't get a Challenge-Response pair in the first place since the PUF never releases responses directly.

⇒ **An extra function that can return responses is needed.**

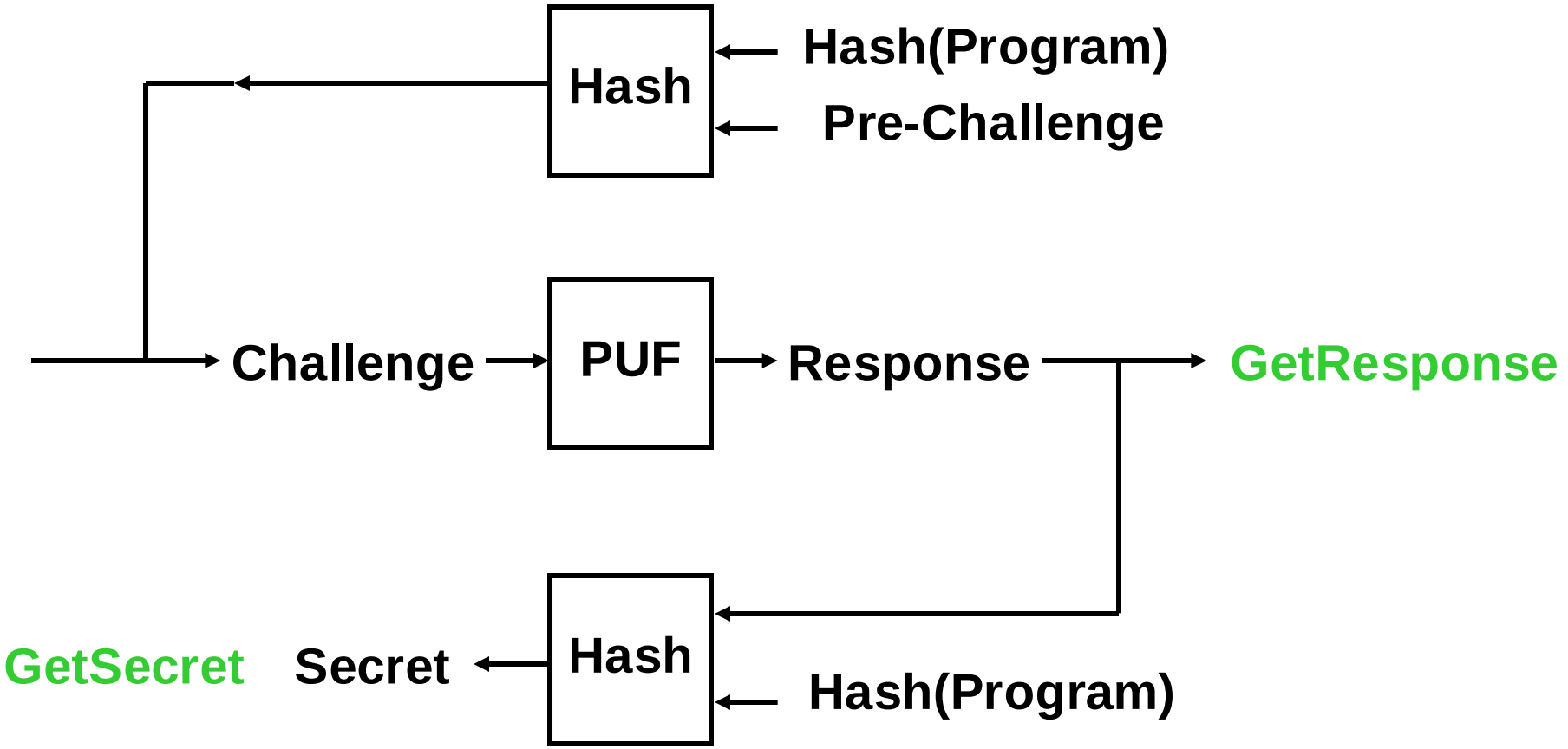
Getting a Challenge-Response Pair – 2

- 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



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

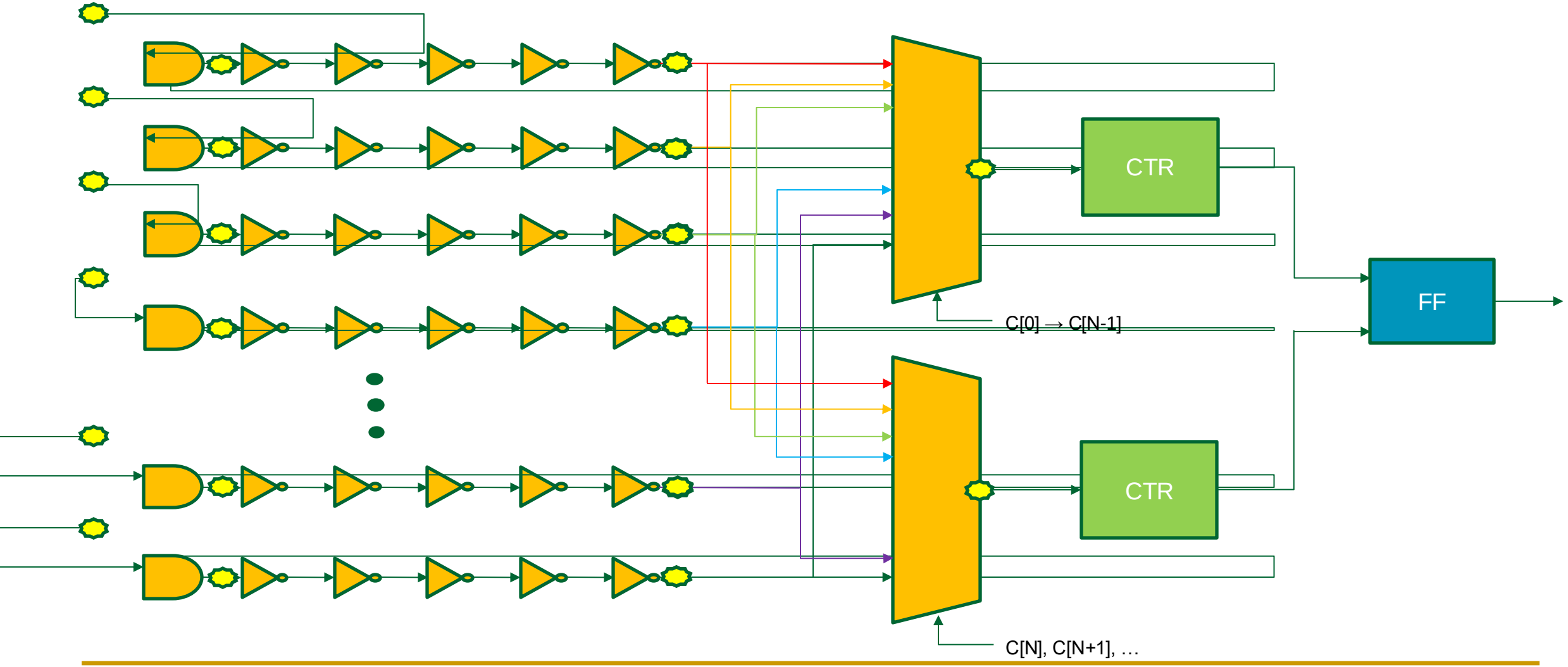
Summary

- 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

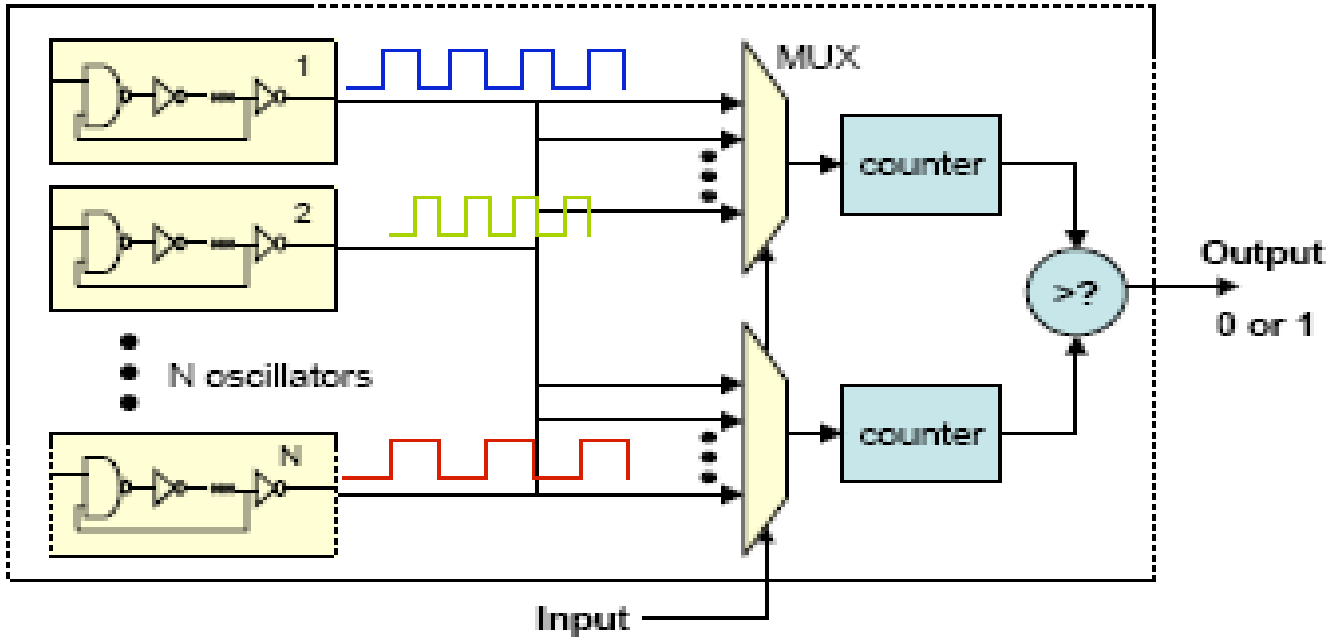
More on Physically Unclonable Functions (PUFs)

Ring Oscillator (RO) PUF



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)

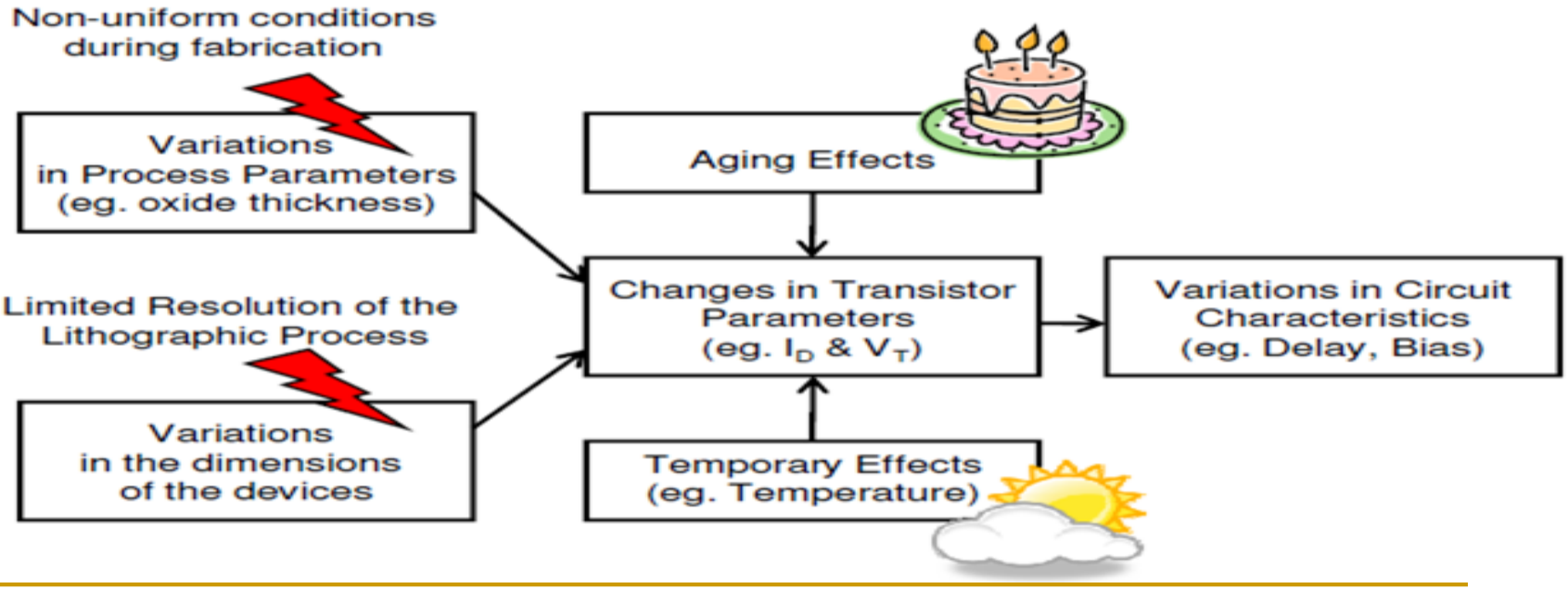
- Easy to duplicate a ring oscillator and make sure the oscillators are identical
  - Much easier than ensuring the racing paths with equal path segments
- How many bits can we generate from the scheme in the previous page?
  - There are N(N-1)/2 distinct pairs, but the entropy is significantly smaller: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

Consider the following minimal example, given three ROs: $RO_A.f < RO_B.f$ and $RO_B.f < RO_C.f$ implicates $RO_A.f < RO_C.f$. The total PUF entropy is only $log_2(N!)$ bit as there are $N!$ ways to sort the frequency values.

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 $$$.



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.