Hardware Security Primitives

Hardware Security Primitives

- Hardware security primitives and designs play an important role to ensure trust, integrity, and authenticity of electronic chips and systems.
- Such primitives can work as the hardware level building blocks to develop a secure platform
- Among common hardware security primitives, physical unclonable functions (PUFs) and true random number generators (TRNGs) are most notable for utilizing device-intrinsic process variations and noise to extract entropy.
- These primitives can be used for generating cryptographic keys and IDs to authenticate devices and systems, to produce session keys, and many more.
- These primitives, acting as a hardware alternative to key storage, digital fingerprint, or software-generated bitstreams can provide defense against prevailing adversarial threats, such as spoofing and cloning.

Hardware Security Primitives

- In addition, researchers have proposed designs for countermeasures, that can defend against IC counterfeiting, tampering, and reverse engineering, by utilizing a different set of device-intrinsic properties.
- For example, combating die and IC recycling (CDIR) sensor leverages aging and wear-out mechanisms in common CMOS-devices to offer countermeasures against IC counterfeiting (recycling)

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.

What We Want to Achieve?

Authentication
no spoofing

Data Integrity
no data alteration

Privacy
no eavesdropping

Attacks

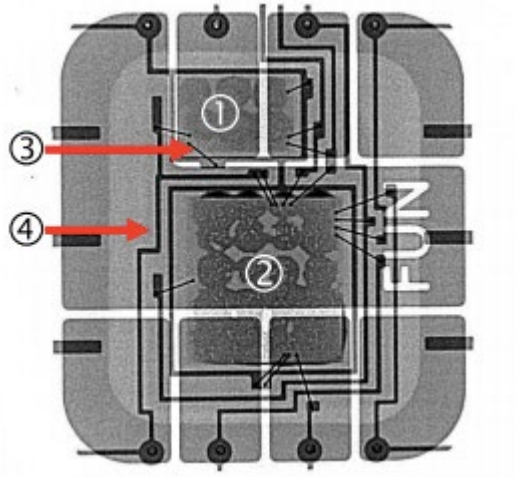
Software-only protection is not enough. Non-volatile memory technologies are vulnerable to invasive attack as secrets always exist in digital form

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.

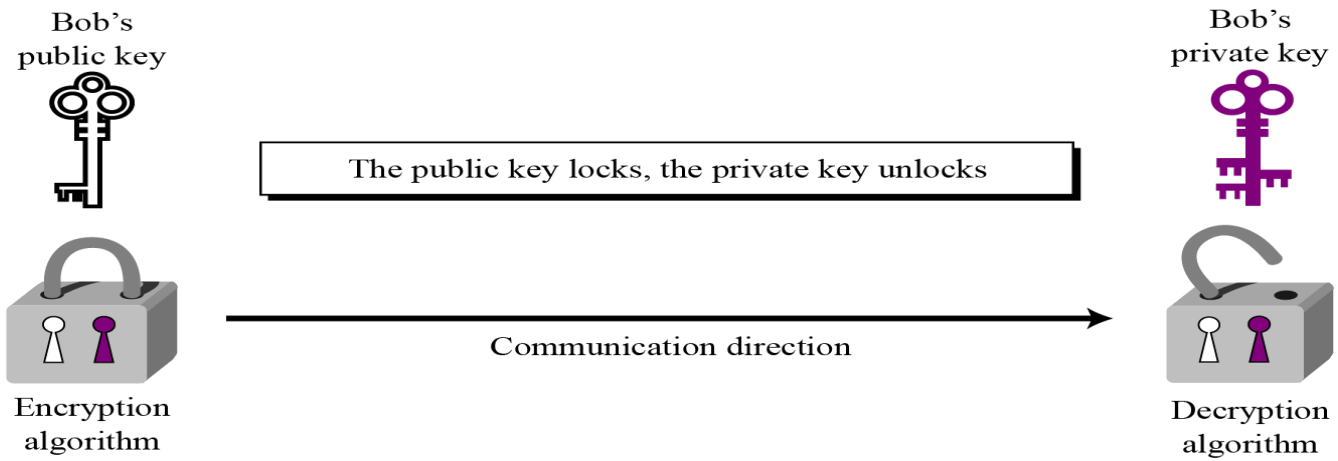
https://arstechnica.com/tech-policy/2015/10/how-a-criminal-ring-defeated-the-secure-chip-and-pin-credit-cards/

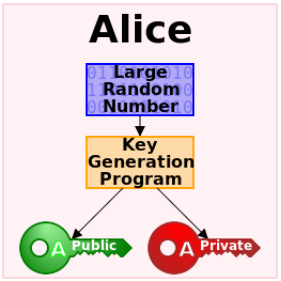


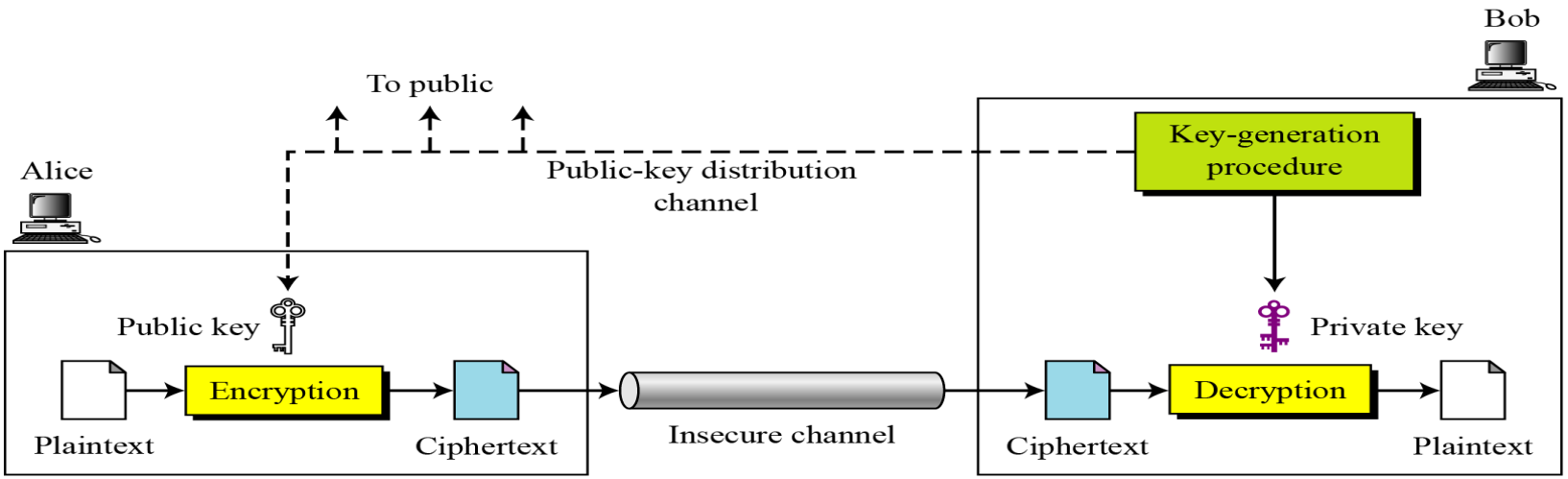
Basic Terminologies

- 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**
  - 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 is still possiple
- **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 are challengeable.

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

IBM 4758

Problem:

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

IBM 4758

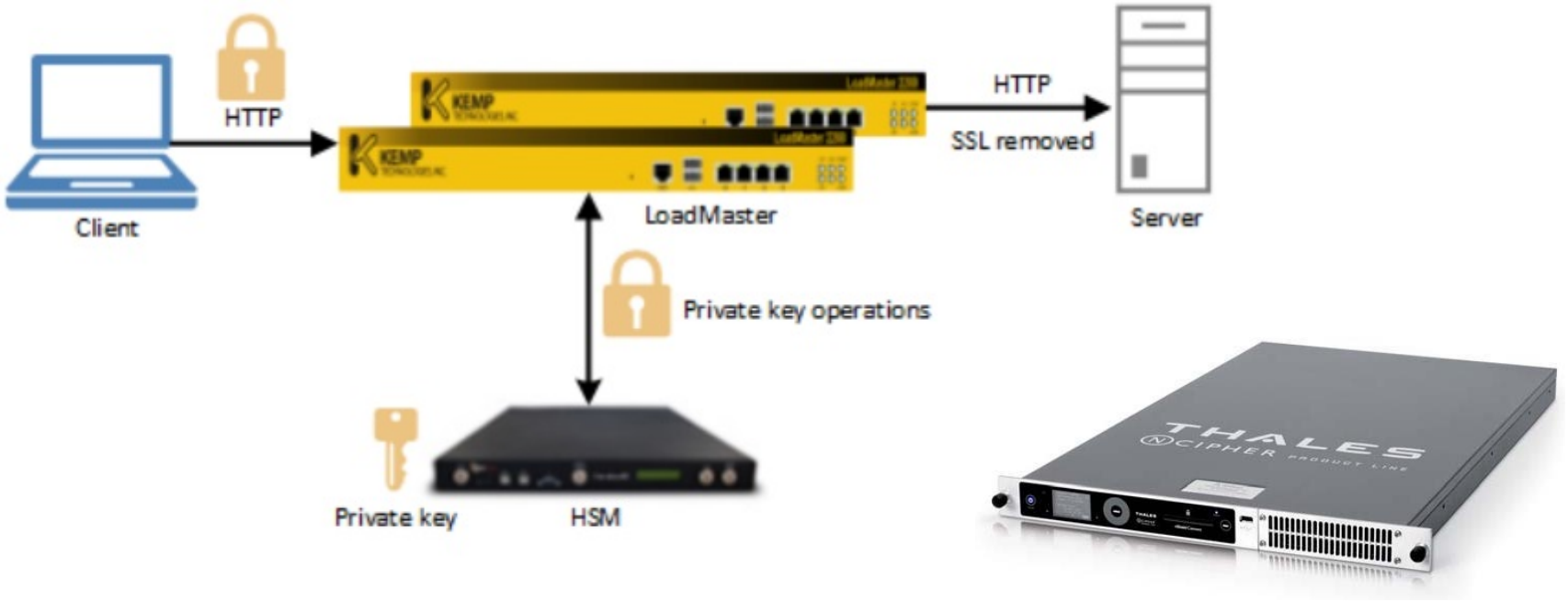
Tamper-proof package containing a secure processor which has a secret key and memory

- Tens of sensors, resistance, temperature, voltage, etc.
- Continually battery-powered

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

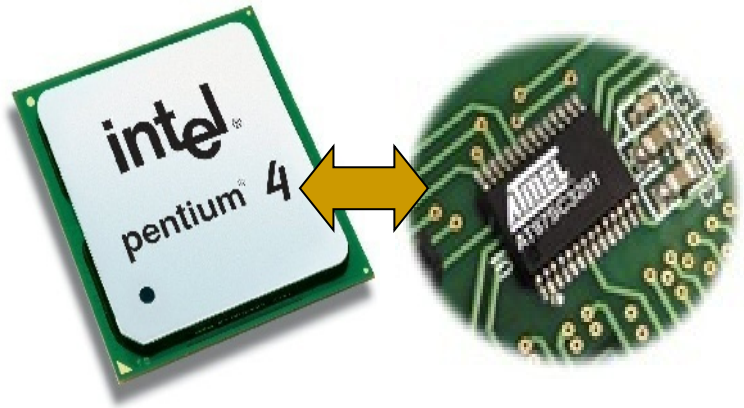
HSM

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

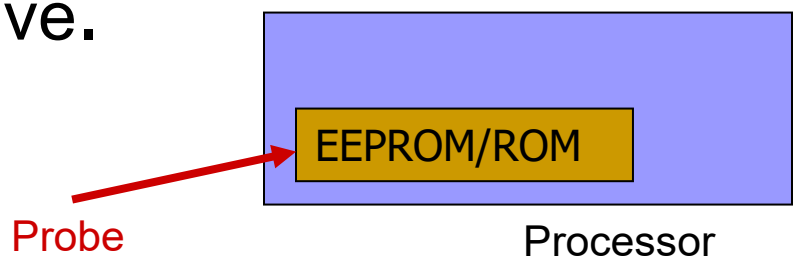


A separate chip (TPM) for security functions

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

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

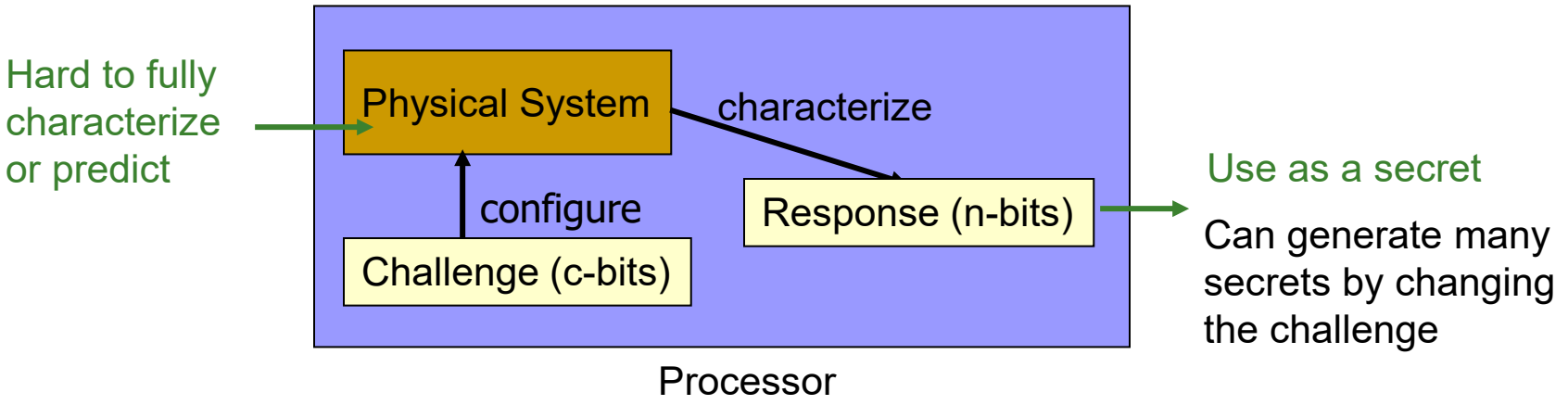
Potential use of inherent device properties for security applications.

CMOS Bulk, HK+MG and FinFET Device Properties for Security Applications

Phenomena		Electrical Manifestation (Performance)	Security Applications	
Process Variations	• Geometric Variation (Patterning – W, L) • Random Dopant Fluctuation (RDF) • Line Edge Roughness (LER) • Oxide Thickness (T_{ox}) Fluctuation • Interface defect and traps (ITC) • Polysilicon/Metal Gate Granularity (MGG)	• Threshold Voltage Deviation (ΔV_{TH}) • Carrier Mobility Degradation ($\Delta\mu_n$) • Drain Current Variation (ΔI_{ON}) • Off-state Leakage Current Variation (ΔI_{Leak}) • Drain Induced Barrier Lowering (DIBL)	PUF	• Arbiter • RO • Leakage Current • Bistable Ring • Hybrid Delay/ Cross-coupled PUF
Aging & Wear-out Mechanisms	• Bias Temperature Instability (NBTI/PBTI) • Hot Carrier Injection (HCI) • Time Dependent Dielectric Breakdown (TDDB) • Electromigration (EM)		TRNG	• Thermal/Power Supply Noise • Clock Jitter • Metastability • Oxide soft-breakdown
Runtime Variations	• Power Supply Noise • Temperature Variation		DfAC	• Recycling – Aging (CDIR) • Cloning – Process Variation

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?

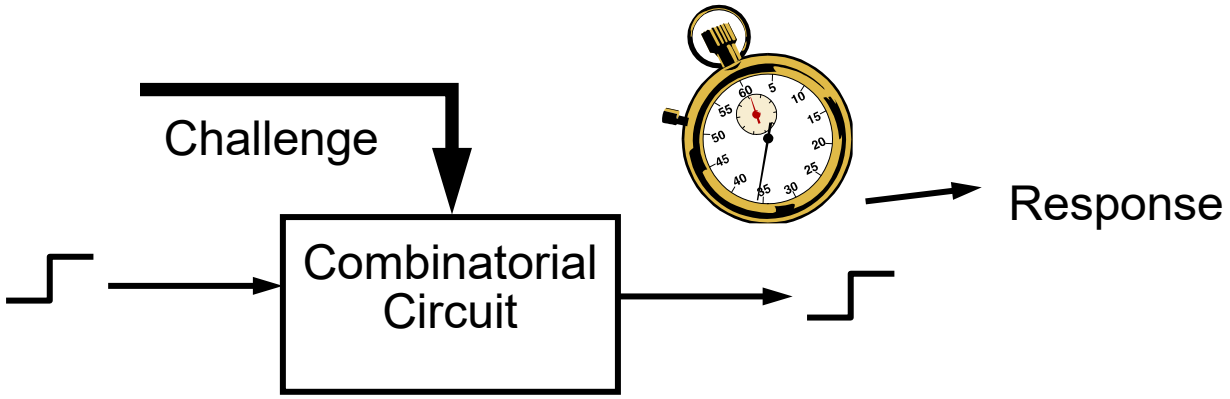
PUF Definition

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

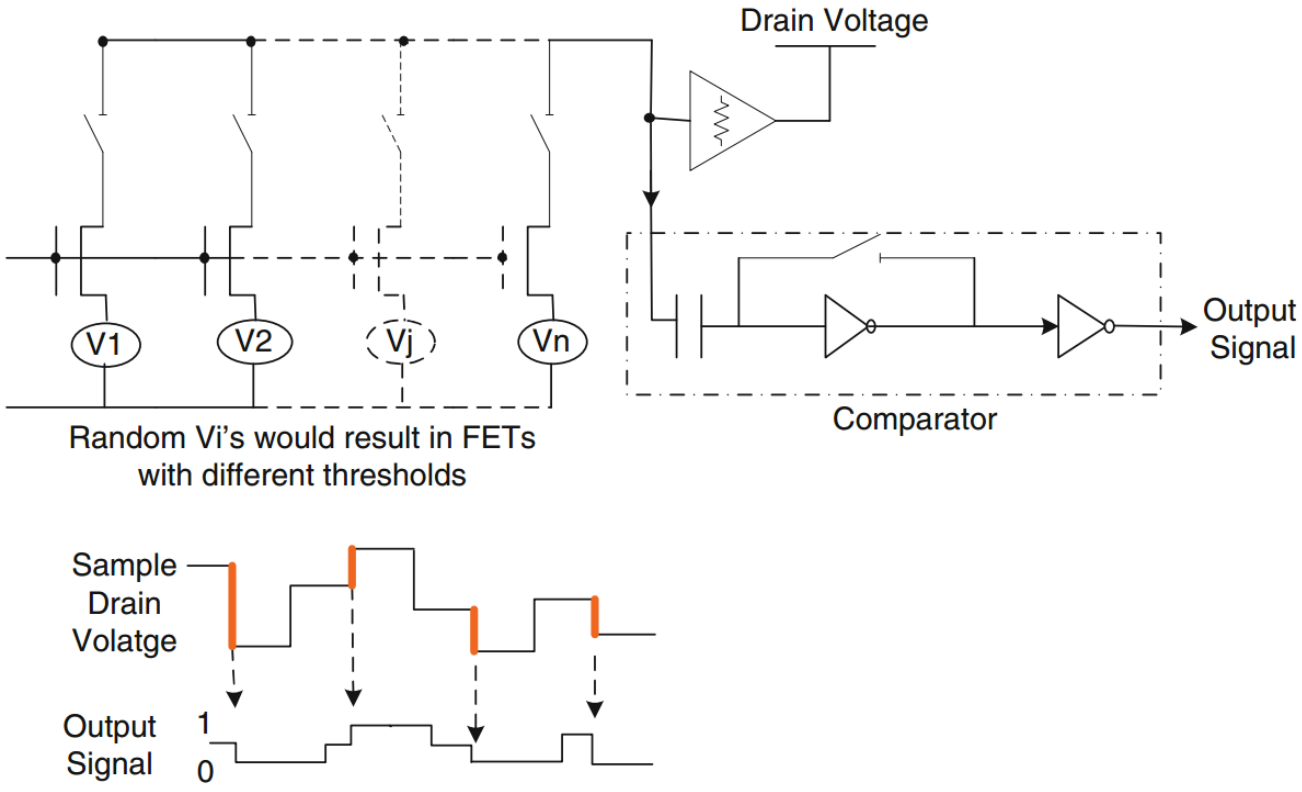
- A digital fingerprint (IDs or Keys)
- Based on a physical system
- Easy to evaluate (using the physical system)
- Its output looks like a random function
- Unpredictable even for an attacker with physical access

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.



Integrated Circuit Identification (ICID)



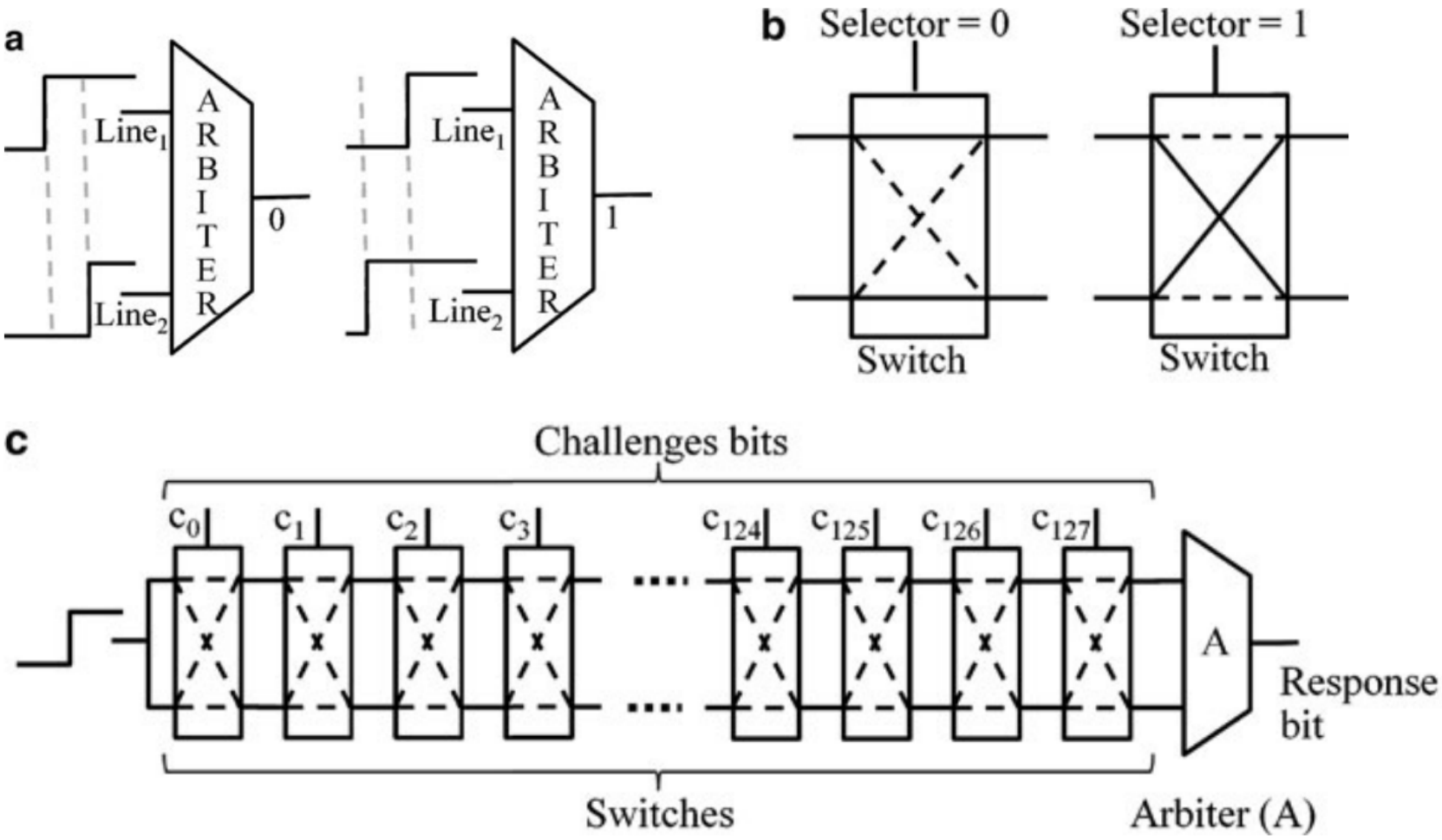
Variable threshold voltage (resulting from process variation) mismatches the drain currents. Therefore, a unique sequence of random voltages would be generated at the load.

Physical Unclonable Function (PUF)

- A PUF is a physically- defined “digital fingerprint” that serves as a unique identity for a semiconductor device such as a microprocessor.
- It is based on unique physical variations which occur naturally during semiconductor manufacturing.

Arbiter PUF

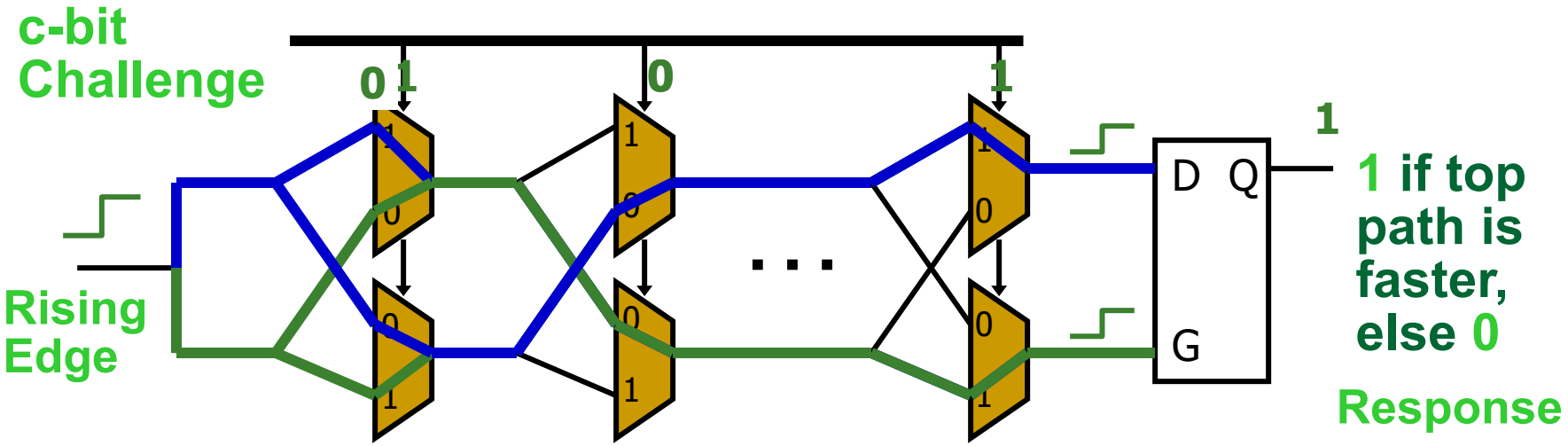
Arbiter-PUF is one of the most notable CMOS logic-based PUF architecture that exploits the randomness of path delay due to uncontrollable process variation.



Arbiter PUF Advantages and Drawbacks

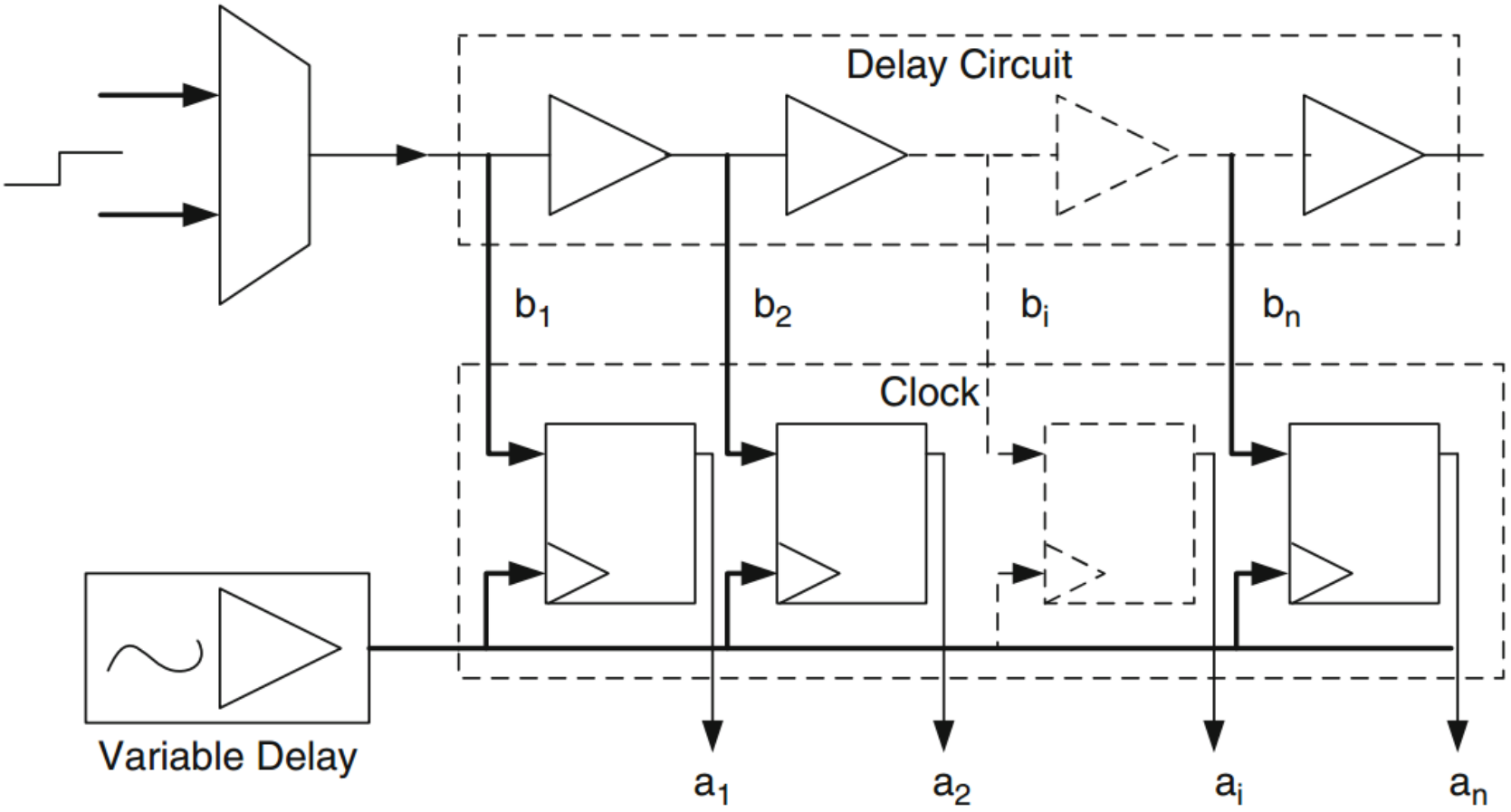
- Advantages:
  - Any random deviation in physical or electrical properties would cause nondeterministic variation.
  - As an increase in the number of delay stages (cascaded one after another in series) exponentially increases possible path-pairs, the arbiter-PUF is capable of generating a large number of Challenge Response Pairs (CRPs) (strong PUF).
  - Another advantage of arbiter PUF is that it takes only one cycle to generate a 1-bit response, although
- Drawbacks:
  - The bias induced at the arbiter itself due to finite delay-difference resolution for the setup and hold times.
  - Also, this requires symmetric design and routing, which may not be readily available for lightweight and FPGA applications.
  - Additionally, the arbiter-PUF has been shown to be vulnerable to modeling attack, since it can be represented as a linear delay model.

A Candidate: Silicon PUF

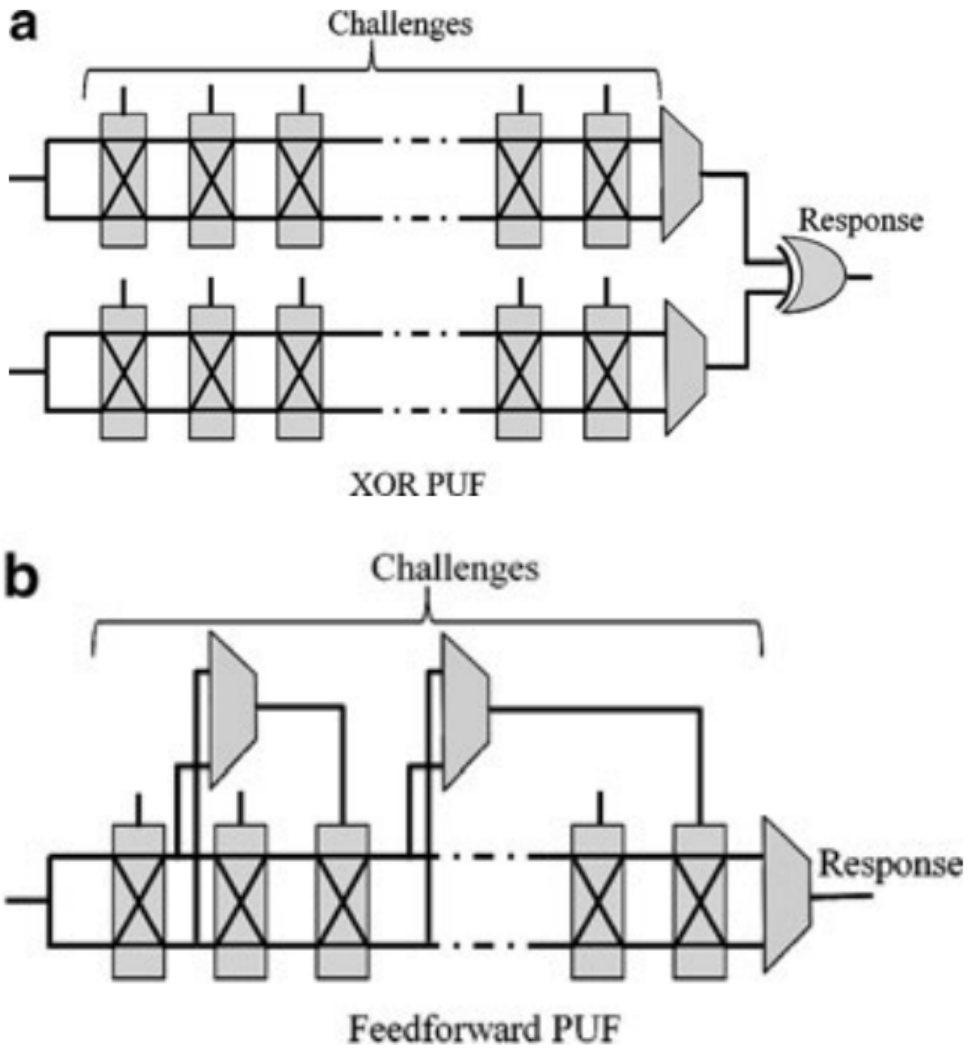


- Compare two paths with an identical delay in design
  - Random process variation determines which path is faster
  - An arbiter outputs 1-bit digital response
- **Path delays in an IC are statistically distributed due to random manufacturing variations**

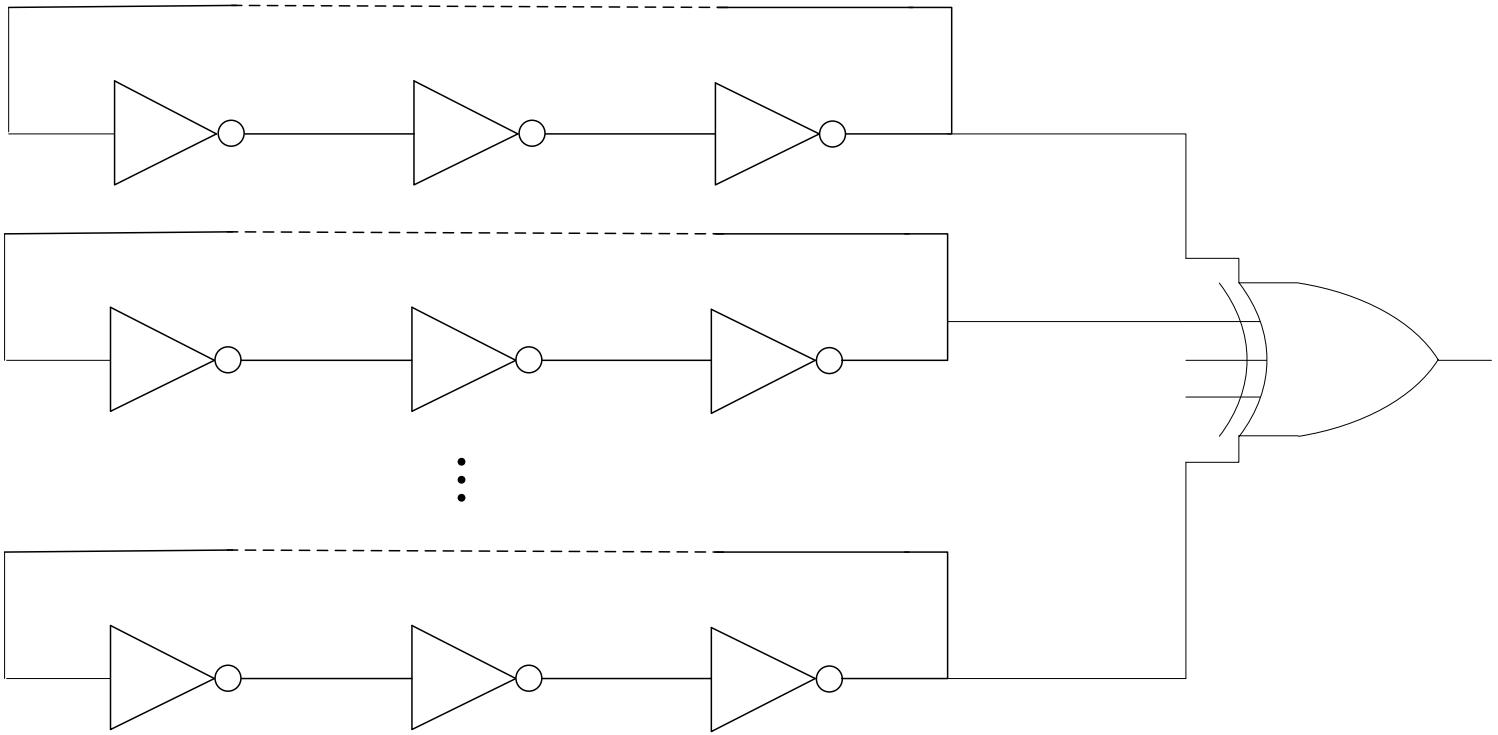
Glitch PUF



Variants of Arbiter PUF

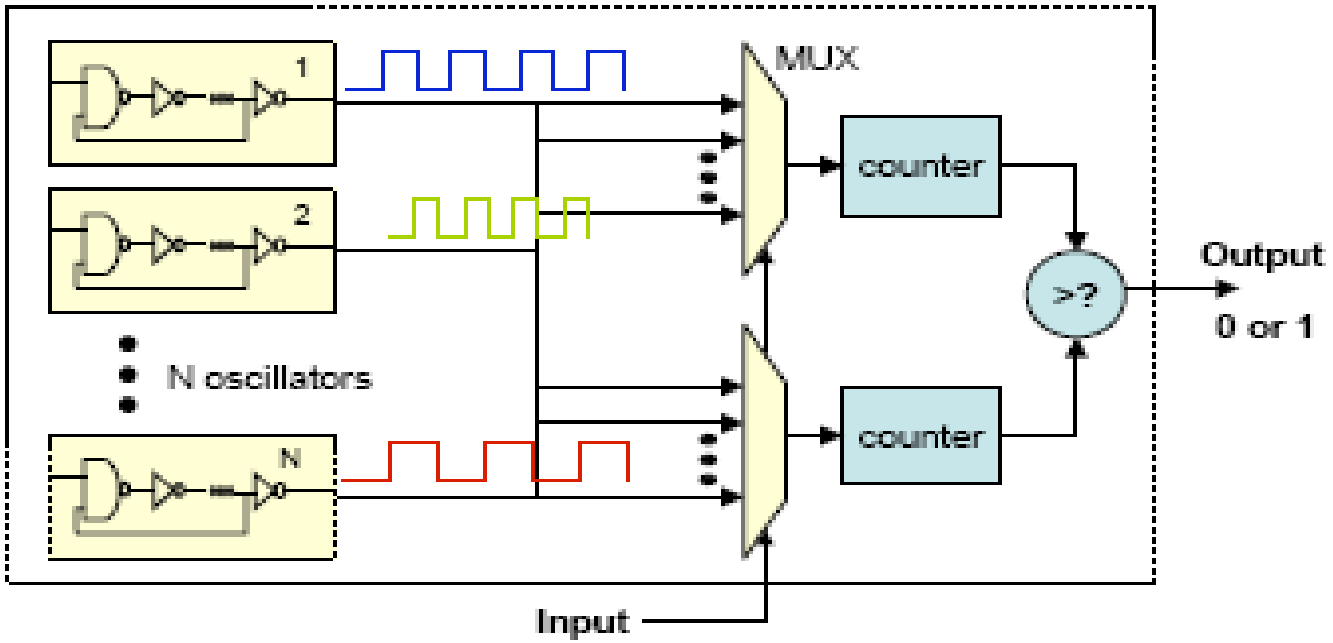


Radom Number Generator (RNG)



Ring-Oscillator (RO) PUF

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

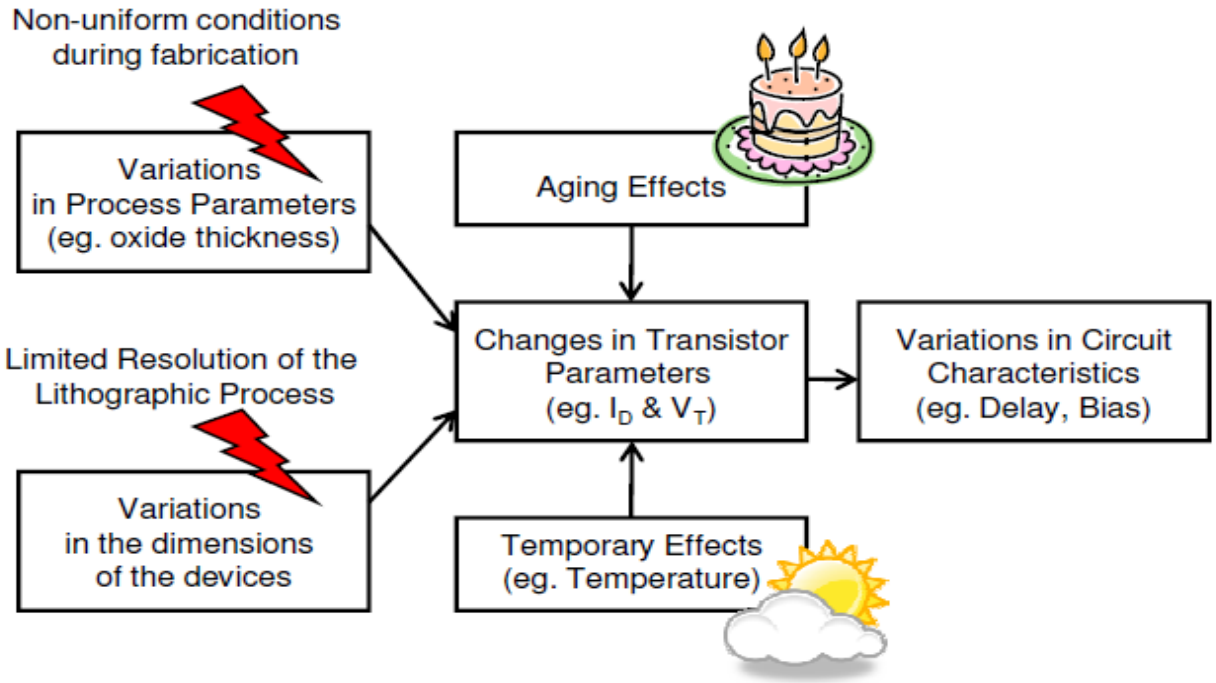


RO PUFs (cont'd)

- 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.

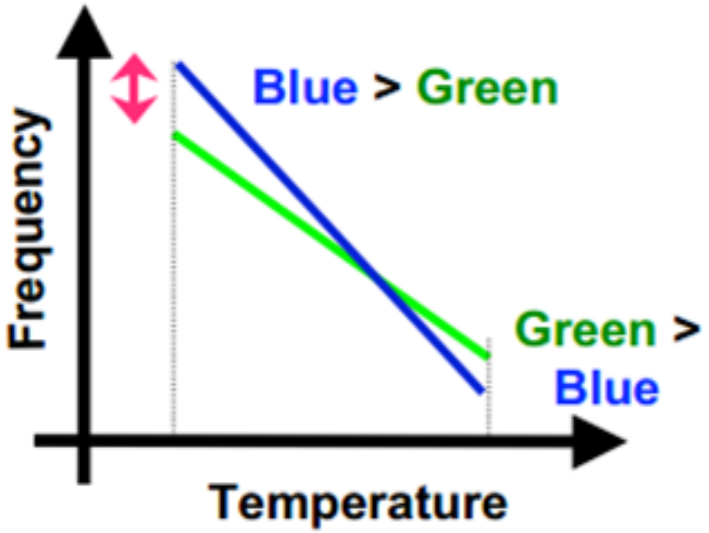
Reliability of RO PUFS



- 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

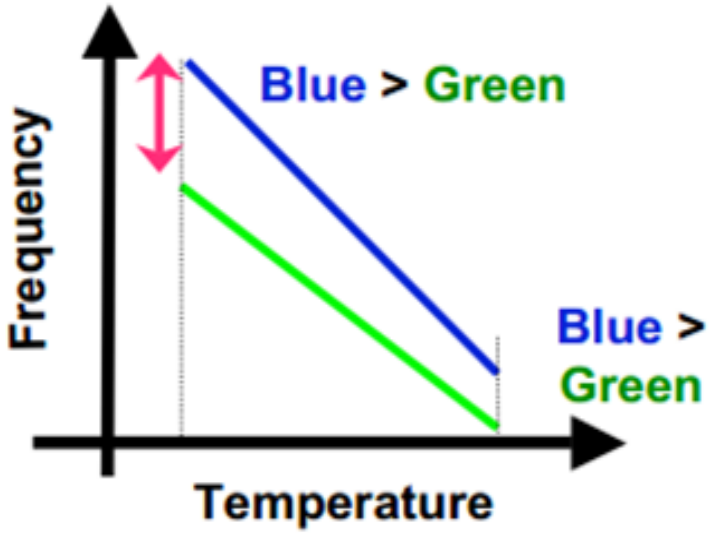
Reliability Enhancement

- Environmental changes have a large impact on the freq. (and even relative ones)
- ROs whose frequencies are far are more stable than the ones with closer frequencies



(a) Frequencies are close

Unstable



(b) Frequencies are far apart

Stable

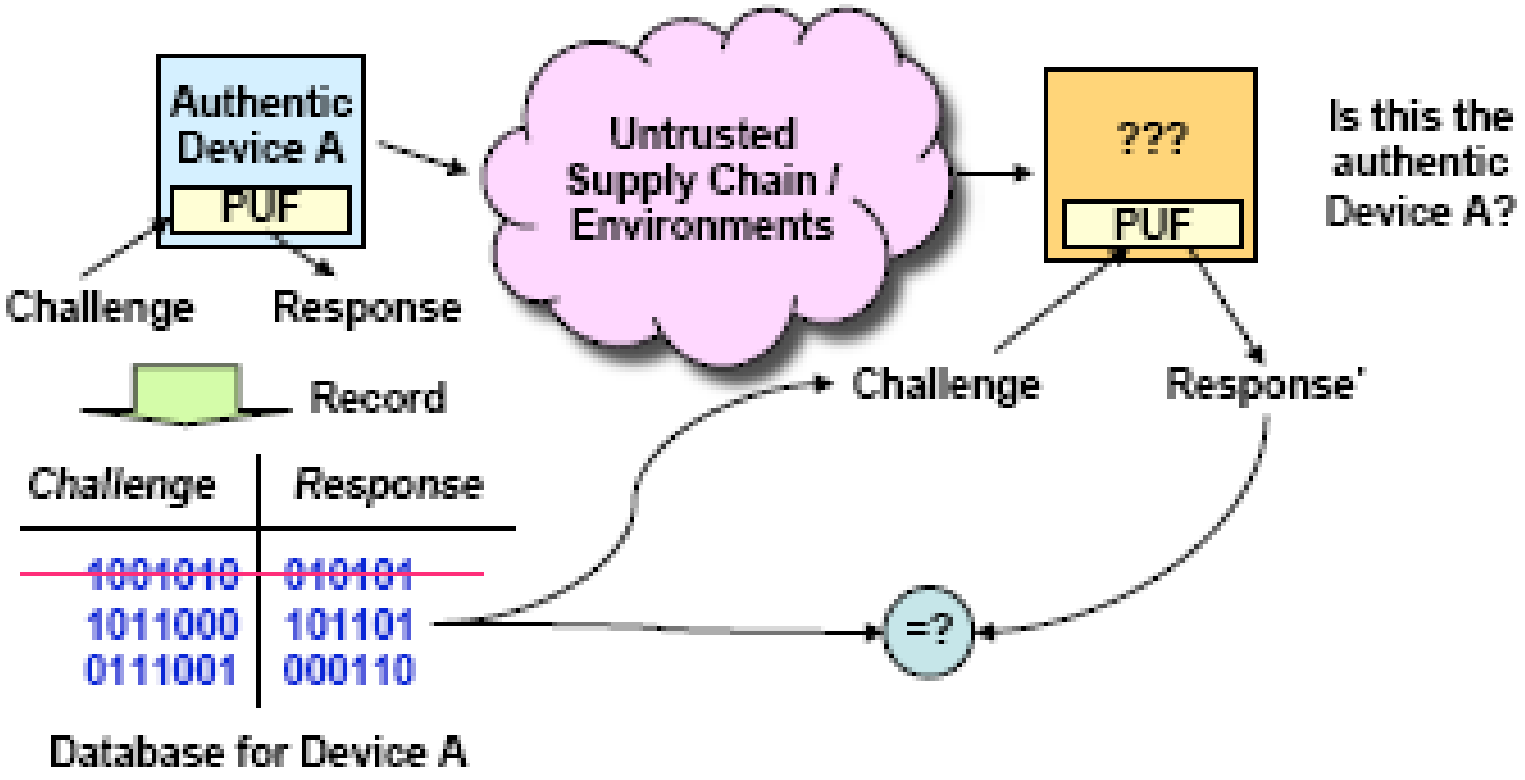
RO PUFs

- ROs usage
  - 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.

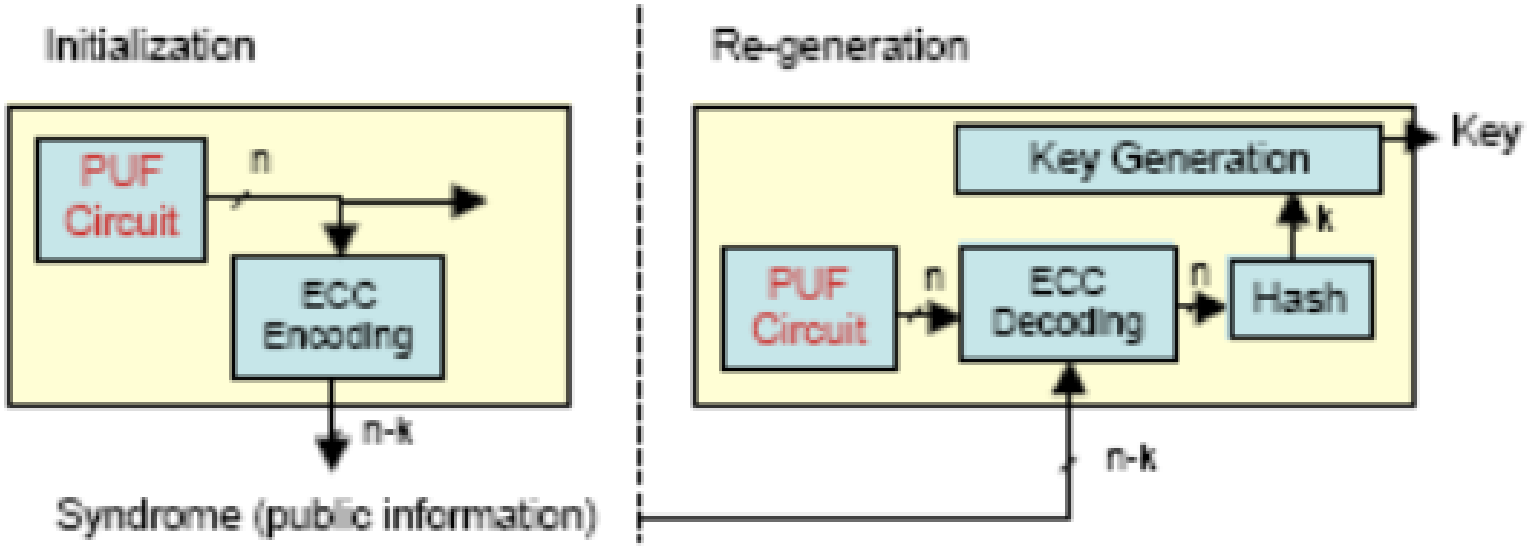
Applications – Authentication

- Same challenges should not be used for different ICs to prevent the man-in-the-middle attacks



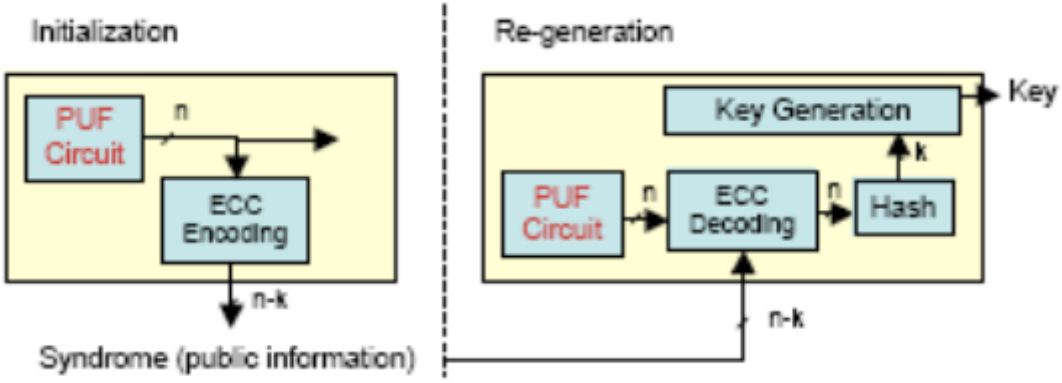
Application – Cryptographic Key Generation

- The instability is a problem because of environmental noise
- 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



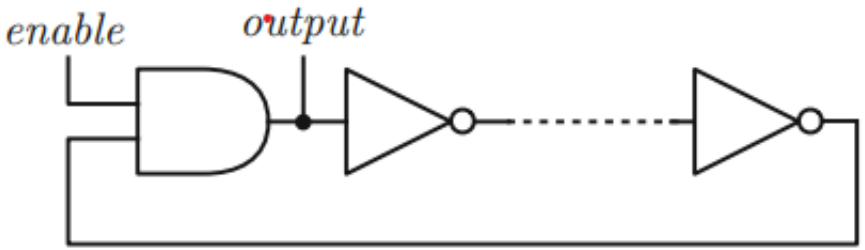
Crypto Key Generation

- Initialization: a PUF output is generated and error correcting code (e.g., BCH) 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



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)

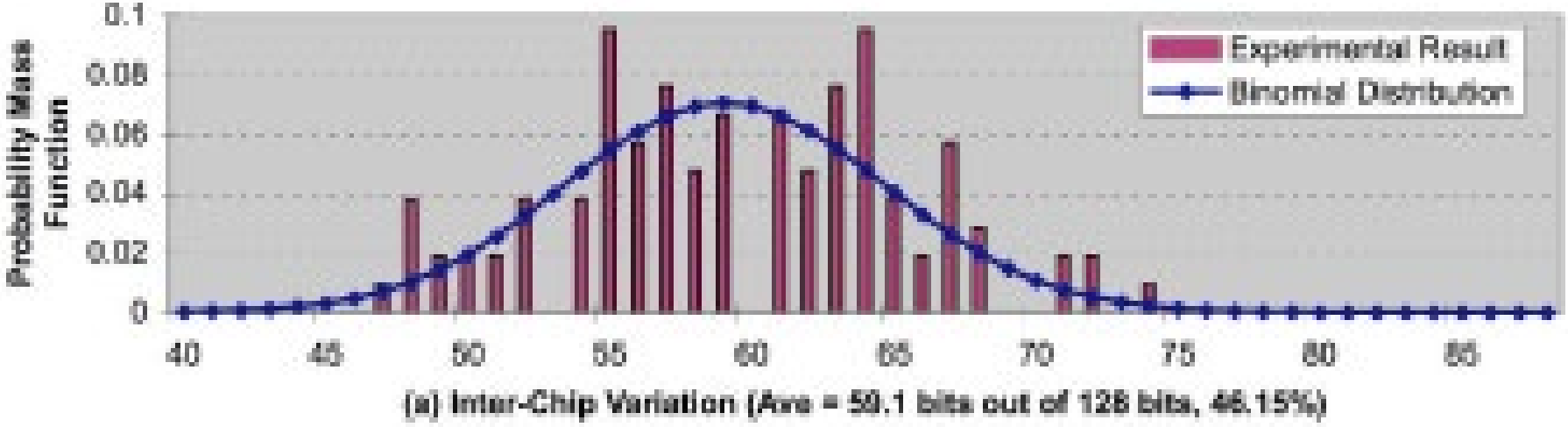


Reliability and Security Metrics

- *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%.

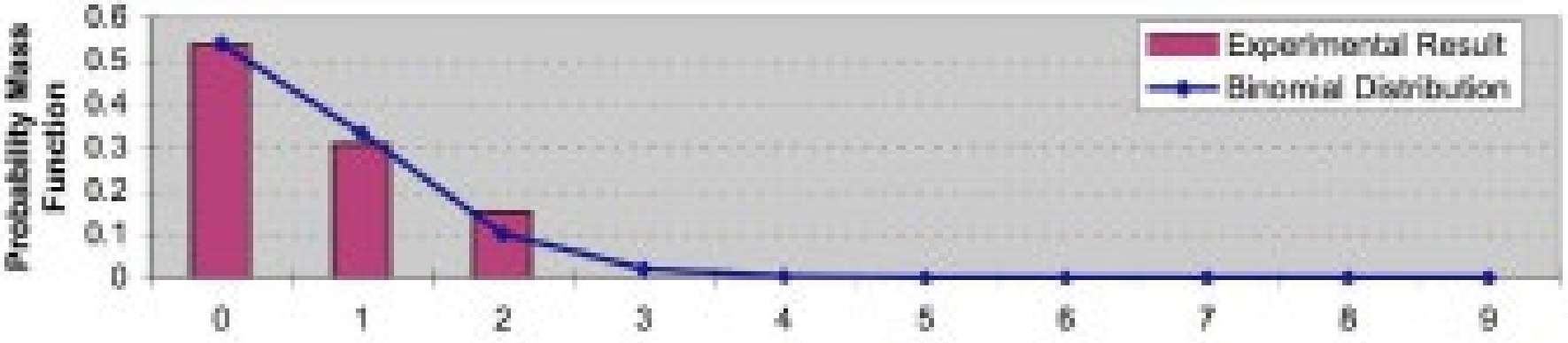
The Probability Distribution for Inter-chip Variations

- 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 0.4615-0.5



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 environmental changes



(b) Intra-Chip Var. under the Worst-Case Env. Change (20C 1.2V vs. 120C 1.08V)
(Ave = 0.61 bits out of 128 bits, 0.48%)

Configurable Ring Oscillator

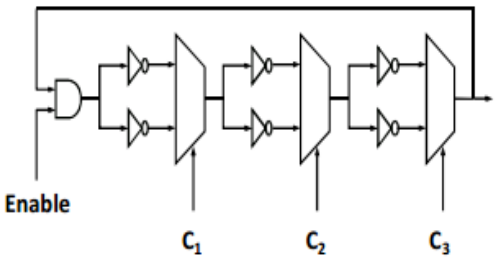


Fig. 3(a)

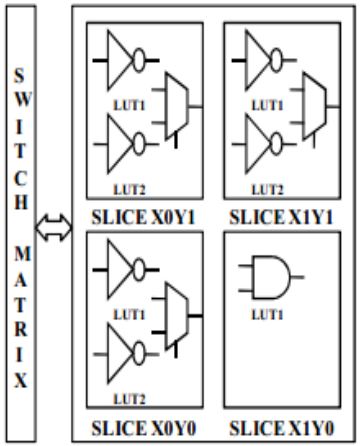


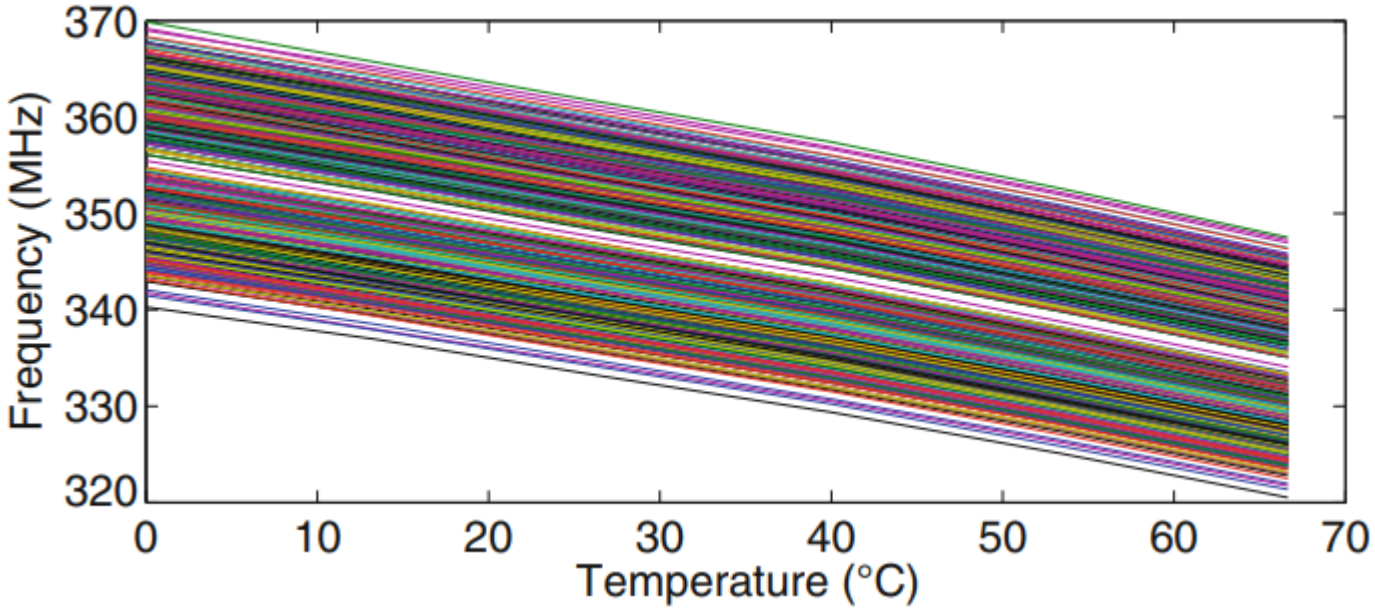
Fig. 3(b)

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	$\lvert f_0 - f'_0 \rvert$
001	f_1	f'_1	$\lvert f_1 - f'_1 \rvert$
010	f_2	f'_2	$\lvert f_2 - f'_2 \rvert$
011	f_3	f'_3	$\lvert f_3 - f'_3 \rvert$
100	f_4	f'_4	$\lvert f_4 - f'_4 \rvert$
101	f_5	f'_5	$\lvert f_5 - f'_5 \rvert$
110	f_6	f'_6	$\lvert f_6 - f'_6 \rvert$
111	f_7	f'_7	$\lvert f_7 - f'_7 \rvert$

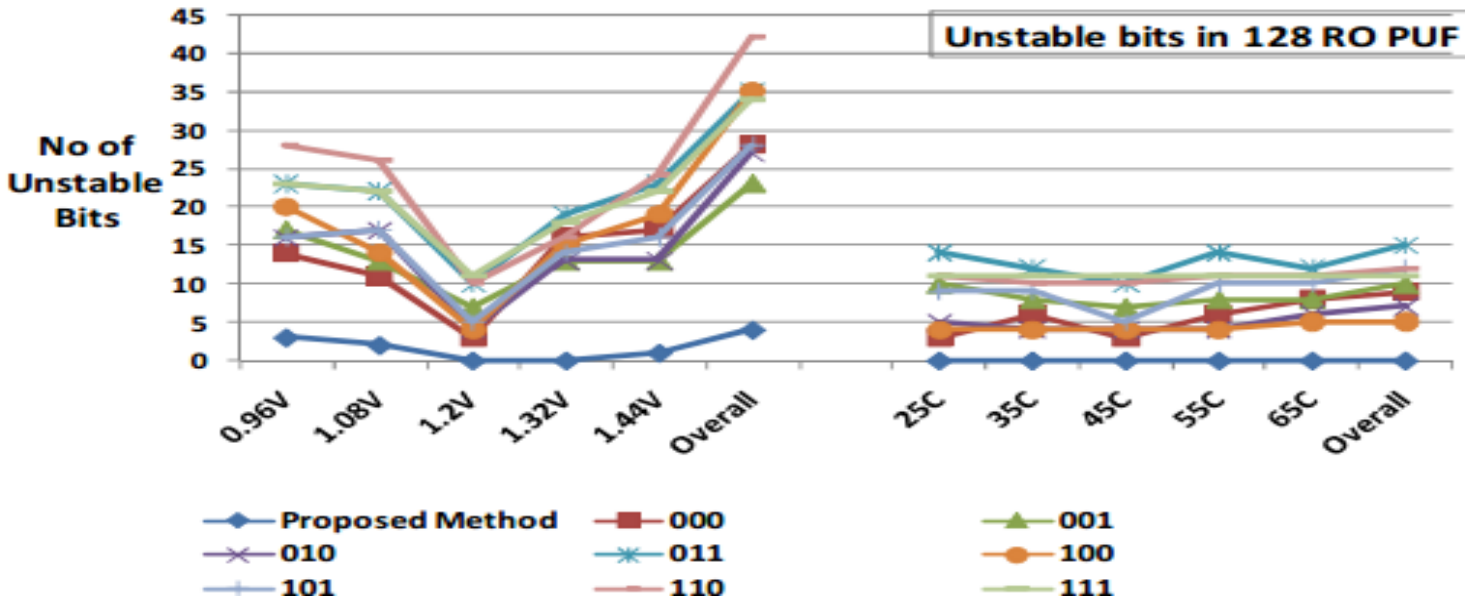
- The pair which has the maximum difference in frequency in CRO is selected.

Configurable Ring Oscillator



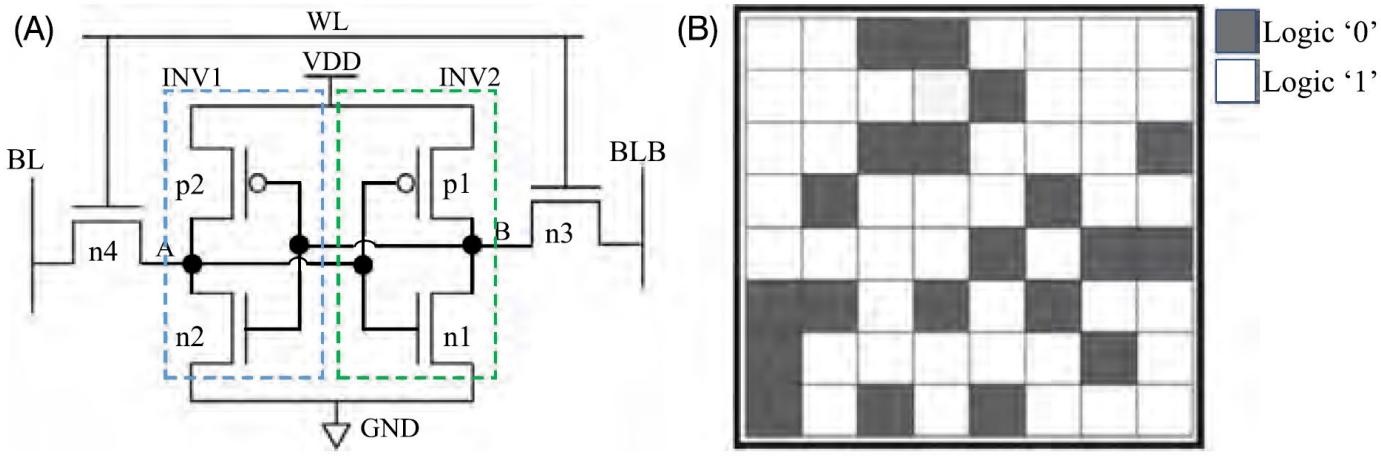
- The pair which has the maximum difference in frequency in CRO is selected.

Configurable Ring Oscillator



- Higher difference in frequency will ensure higher reliability
- Redundancy

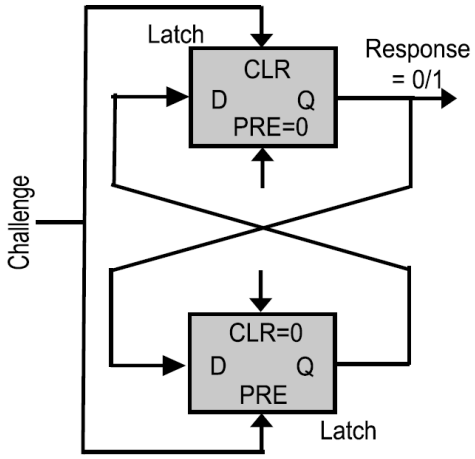
SRAM PUF



(A) Typical 6T SRAM cell. (B) Startup fingerprint of an example SRAM array.

- In contrast to custom-designed arbiter- and RO-PUFs, a static random-access memory (SRAM)-based PUF utilizes the widely available SRAM-matrix used in microprocessors, microcontrollers, field programmable gate arrays (FPGAs) and in standalone chips for embedded systems.
- The process variation in the SRAM cell may be small enough to be overcome by environmental noise and, therefore, not all cells can produce a reliable response over time and usages.
- Additionally, advanced SRAM-inclusive commercial off-the-shelf (CoTs) products may not readily provide suitable PUF application due to various initialization and memory-access processes. For example, in some recent Altera and Xilinx FPGA models, RAM blocks are always initialized to certain logic at startup and, therefore, cannot be used to implement the SRAM-PUF based on a random initialization

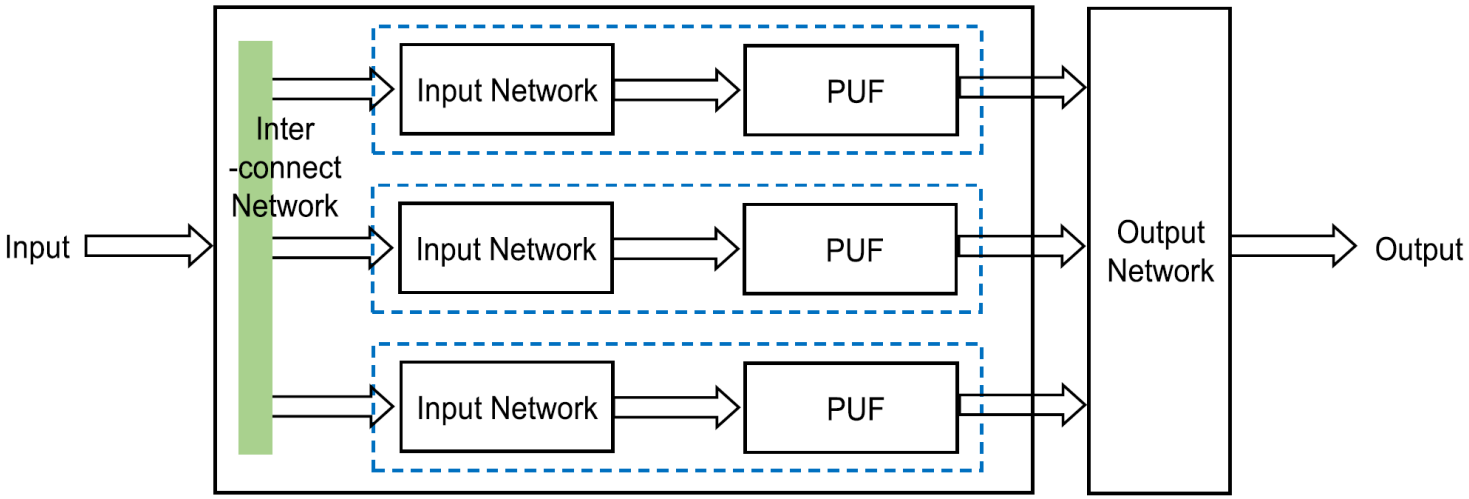
Butterfly PUF



Typical butterfly PUF schematic

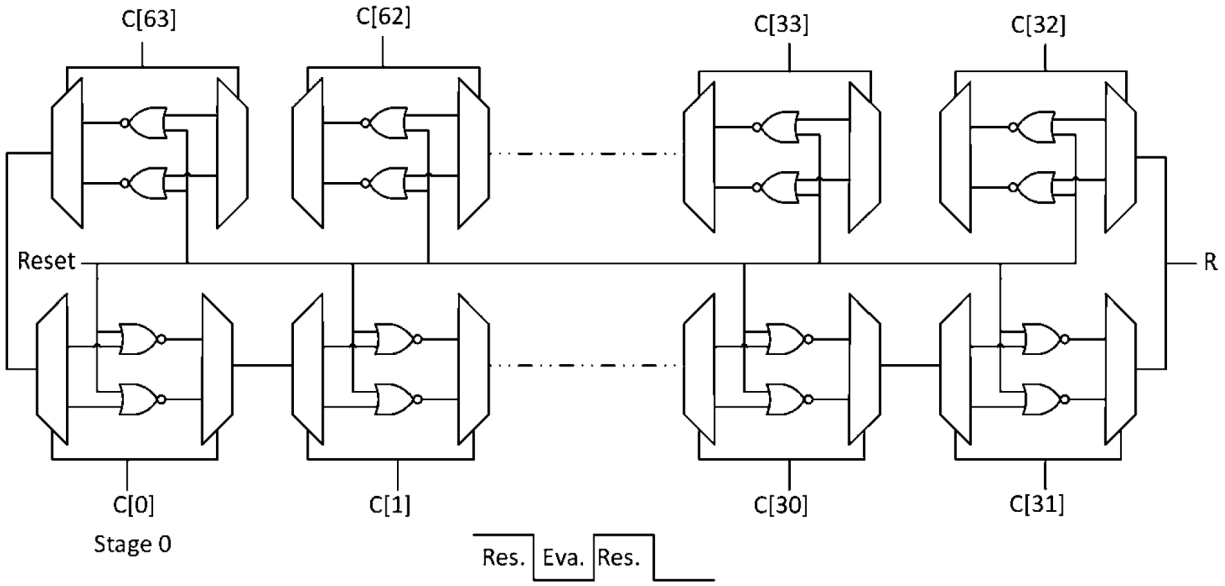
- The design of Butterfly PUF is inspired by the notion of creating a circuit structure with metastable properties in FPGA matrix.
- Similar to the SRAM-PUF, the floating state of Butterfly PUF can be exploited to obtain a random state at the startup phase of a pair of cross-coupled latches in the FPGA
- The latches form a combinational loop that can be excited to an unstable state through a signal, challenge.
- Although conceived for FPGA implementations, similar implementation can be done utilizing the metastability of any latch or flip-flop-based architecture.

Lightweight PUF



- Lightweight PUF utilizes traditional arbiter-PUF with nontrivial wiring between arbiter stages.
- Rather than directly feeding the challenges to the PUFs, it creates a networked scheme that breaks down the challenge set into several blocks, and uses them on multiple individual PUFs.
- The output network then combines all the individual-PUF responses to create a global response, making it more resilient against machine learning attacks.

Bistable Ring PUF



A 64-stage bistable ring PUF

- A bistable ring contains an even number of inverters and can only have two possible stable outcomes.
- However, due to manufacturing process variation and noise, the bistable ring goes through a set of complex transitions (or metastability) before converging to a stable state.
- A bistable ring PUF can exploit the metastability to generate exponential number of CRPs.
- Similar to arbiter PUF, it requires a symmetric layout. Nevertheless, this strong PUF offers relatively high resiliency against modeling attacks due to its complex and nonlinear nature, and thereby can be incorporated into emerging PUF applications such as *Virtual Proof of Reality*

True Random Number Generator

Random Numbers in Cryptography

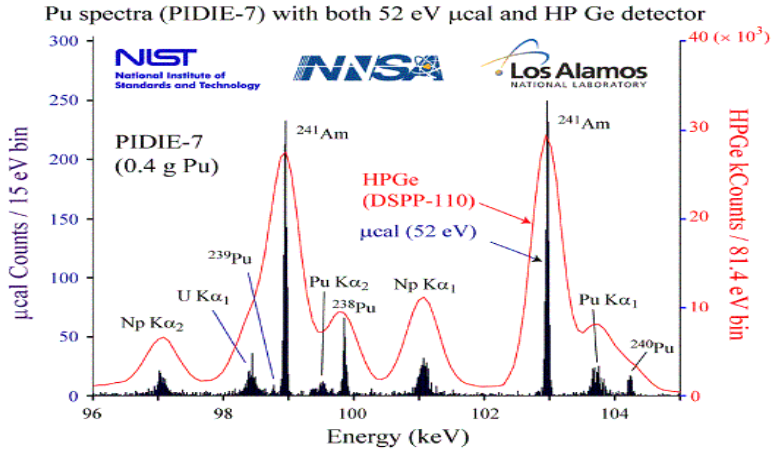
- The keystream in the one-time pad
- The secret key in the DES encryption
- The prime numbers p, q in the RSA encryption
- Session keys
- The private key in digital signature algorithm (DSA)
- The initialization vectors (IVs) used in ciphers

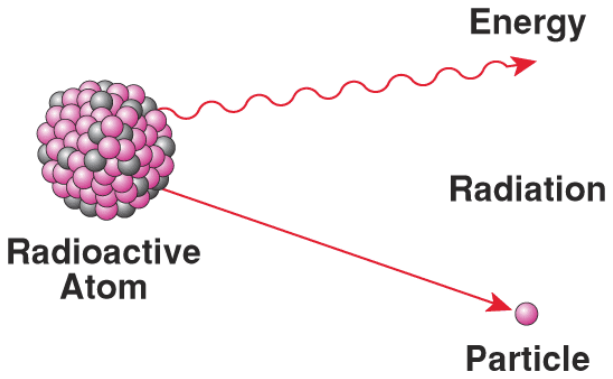
Pseudo-random Number Generator

- **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

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.





- Randomness is bound to natural phenomena. 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.

https://www.aps.org/units/fps/newsletters/2006/july/article6.html

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.

Shot noise, the time-dependent fluctuations in electrical current caused by the discreteness of the electron charge, is well known to occur in solid-state devices.

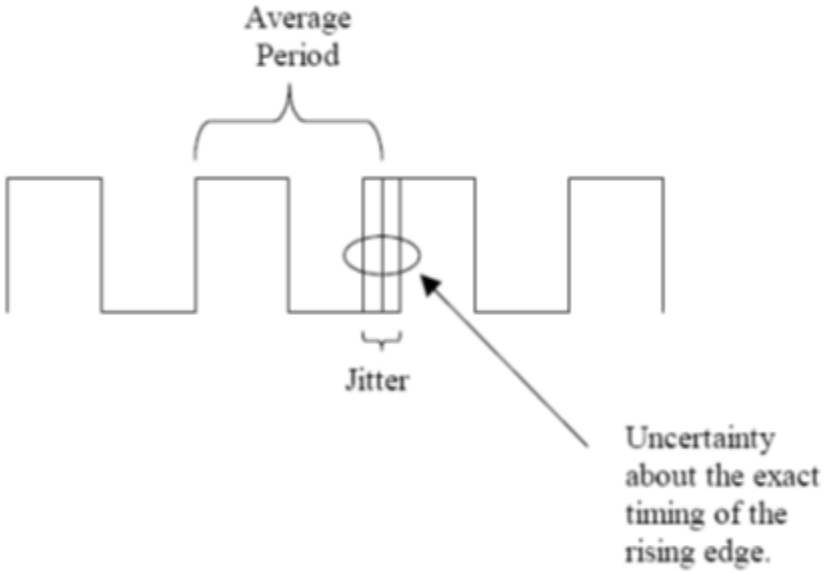
Set of Requirements

- 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

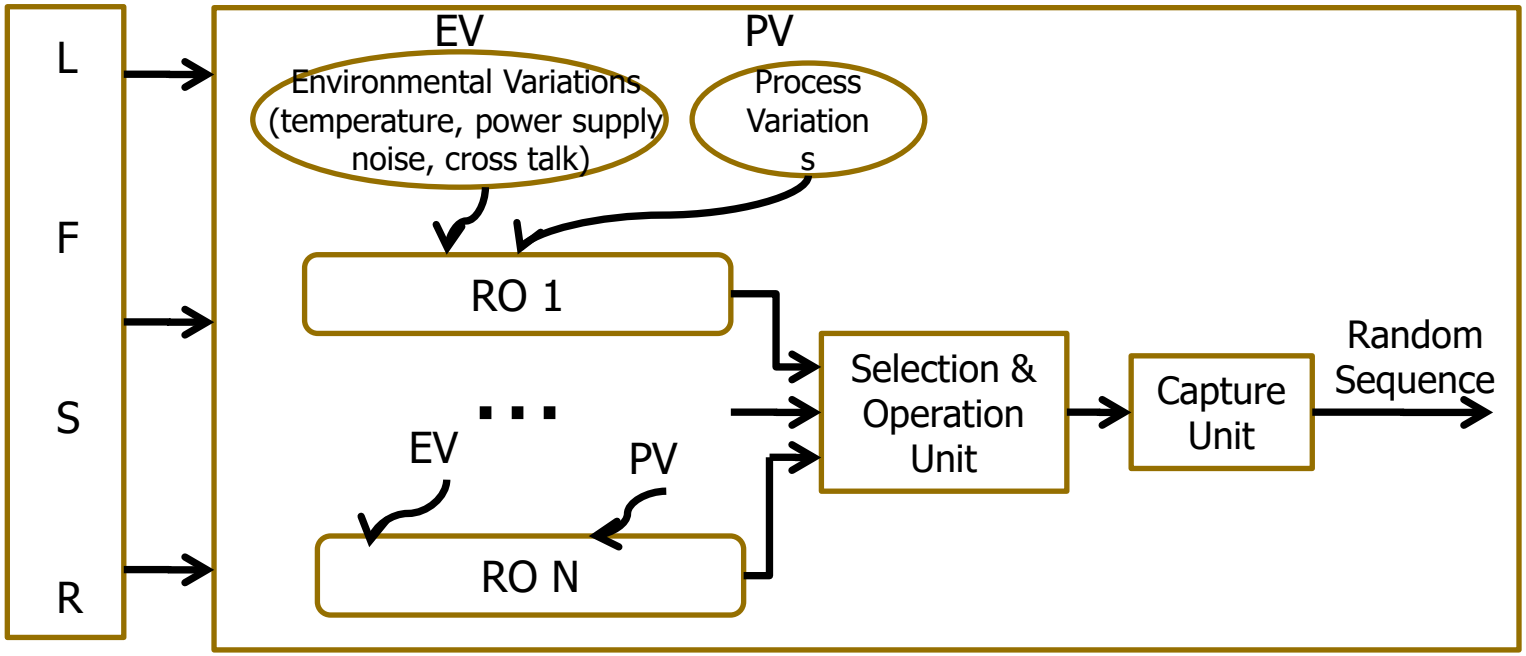
- Jitter is variations in the significant instants of a clock
- Jitter is nondeterministic (random)

- **Sources of Jitter:**
  - Semiconductor noise
  - Cross-talk
  - Power supply variations
  - Electromagnetic fields



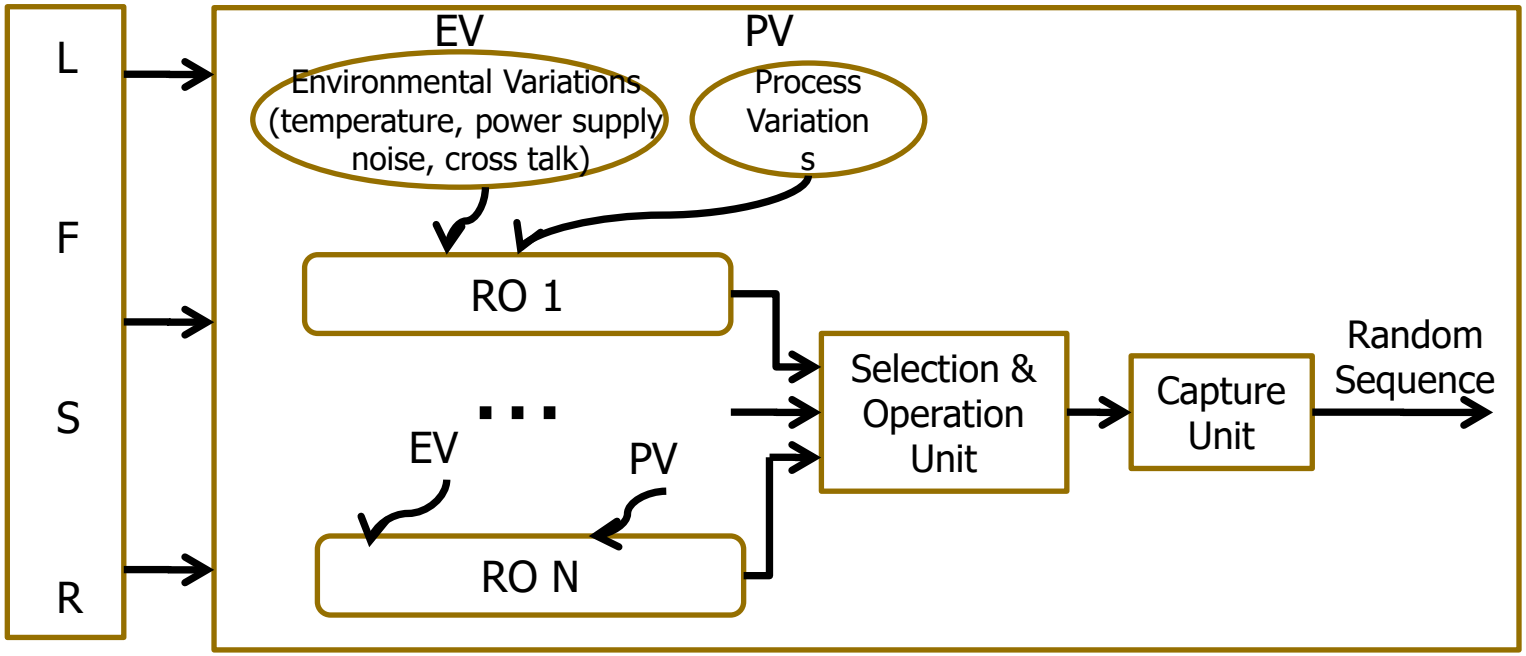
TRNG Structure

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



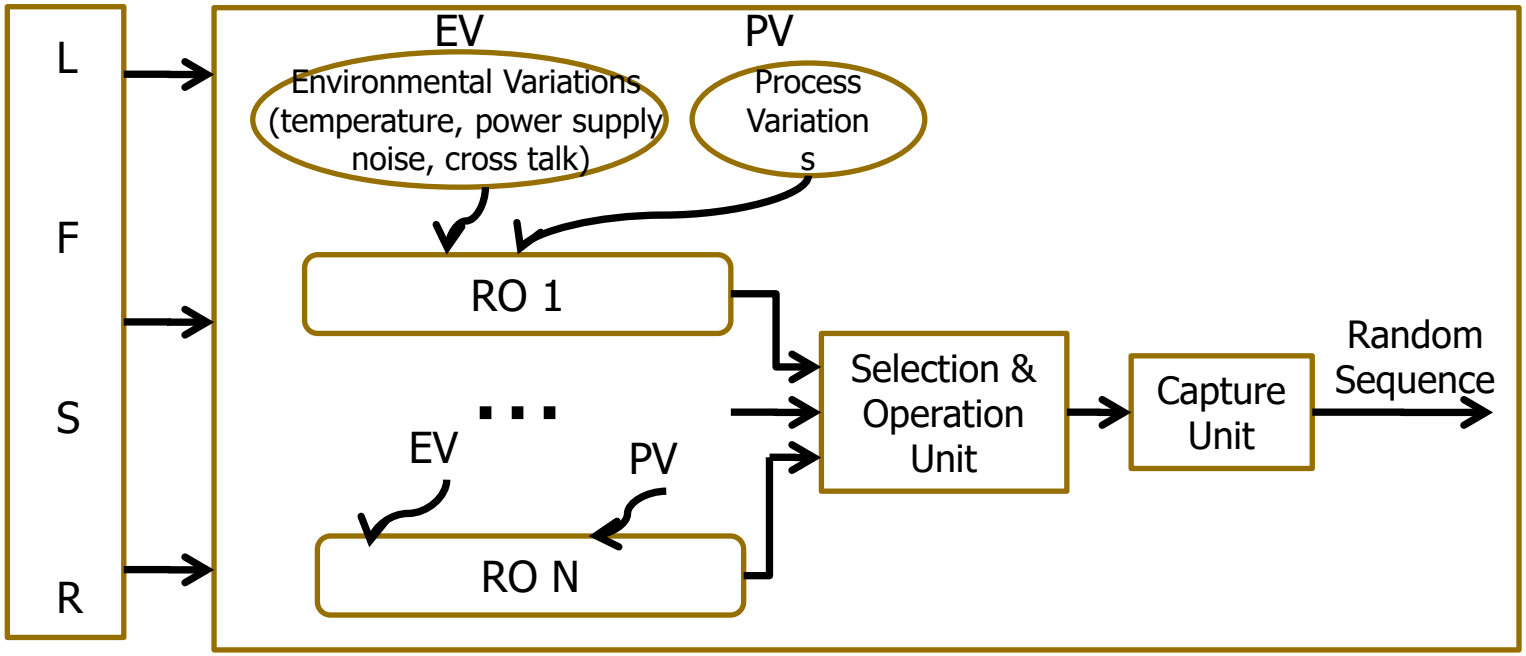
TRNG Structure

- **Ring Oscillators**
  - Process variations & environmental variations
  - Random phase jitter



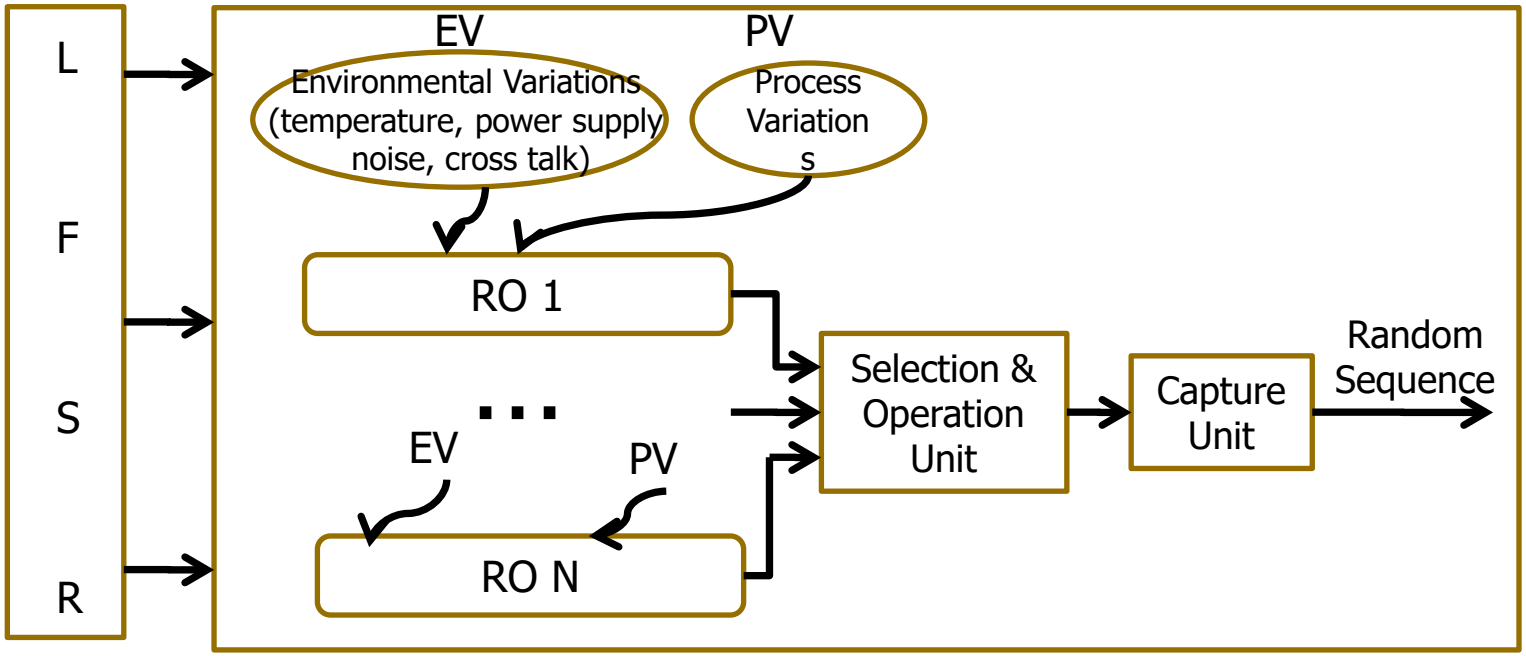
TRNG Structure

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



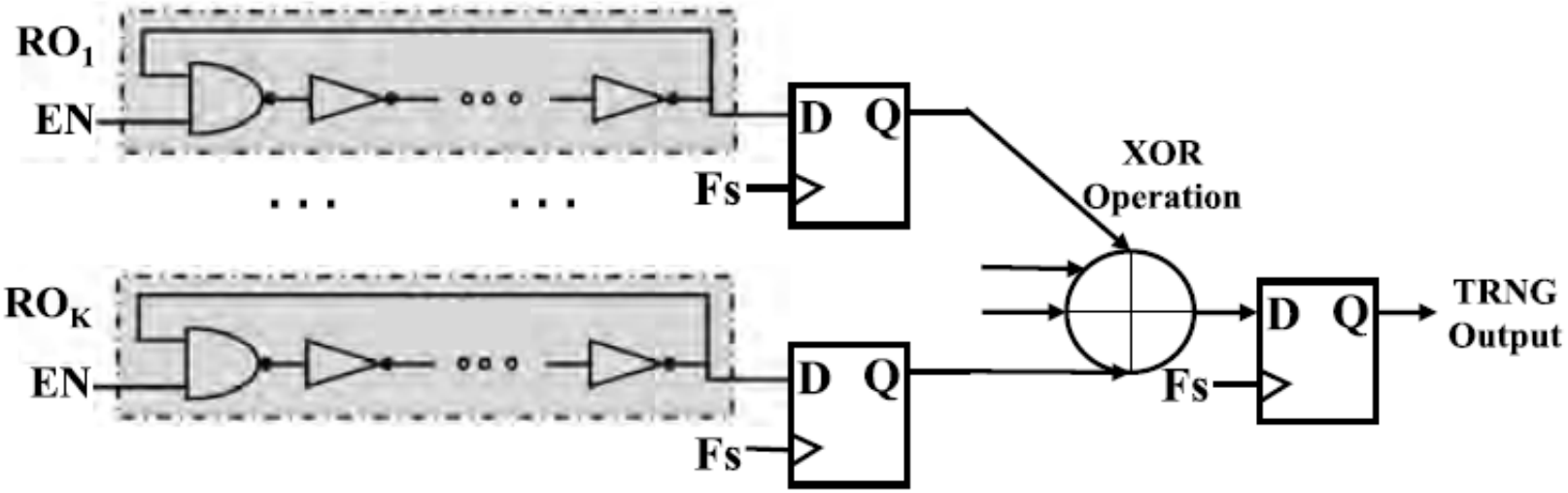
TRNG Structure

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

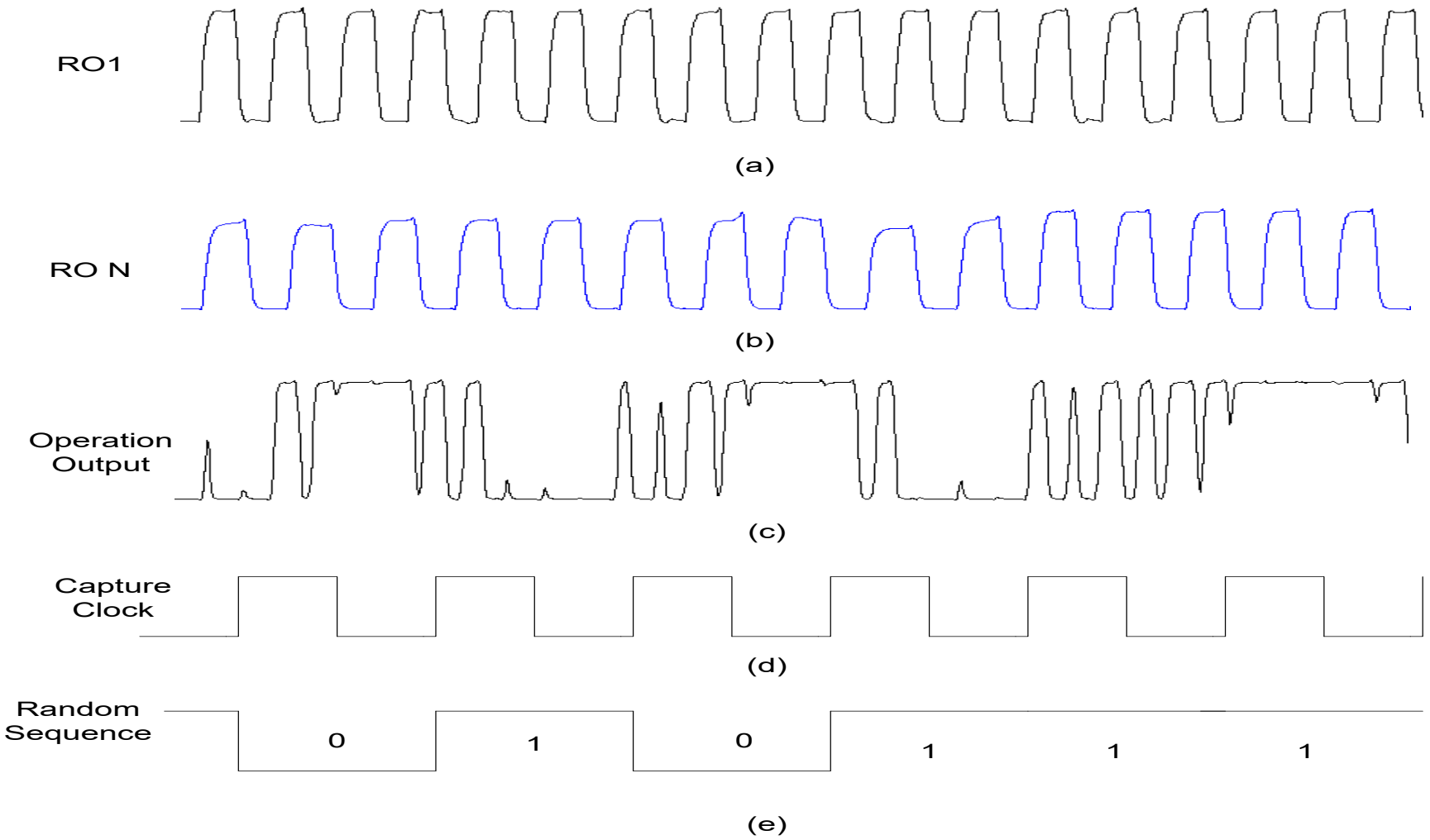


RO-based TRNG Example

- Use oscillators and leverage associated jitter and metastability.
- Odd numbers of back-to-back connected inverters, with a feedback loop, act as a free-running oscillator (FRO).
- Random electrical noise in the feedback loop causes the frequency and phase of the oscillation to have jitter, that is, the exact time of the signal reaching the extraction point is not deterministic.
- The entropy is further improved by proper sampling and XORing each FRO output.

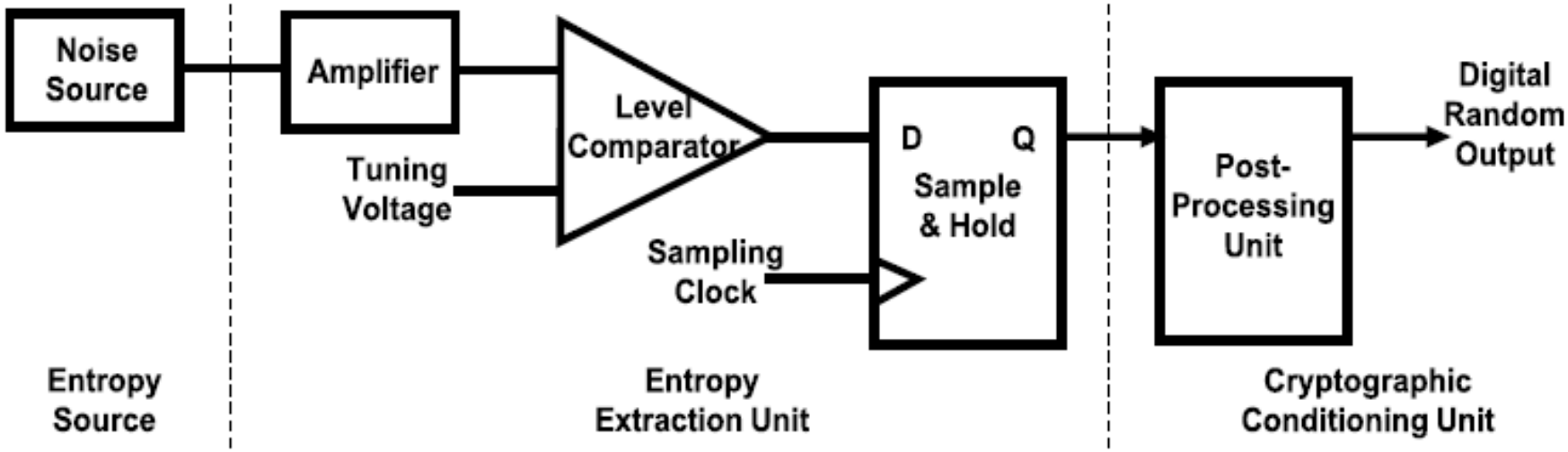


TRNG Output


RO1
(a)
RO N
(b)
Operation Output
(c)
Capture Clock
(d)
Random Sequence
0
1
0
1
1
1
(e)

Noise-based TRNG

- Device-inherent noise is typically random and can be harnessed for generating true random numbers.
- The basic idea of noise-based TRNG is as follows: the random analog voltage, caused by the noise source, is sampled periodically and compared to a certain pre-defined threshold to produce a “1” or “0”. The threshold can be fine-tuned to produce an ideally equal probability of “1”s and “0”s.
- However, setting up a proper threshold may be a rigorous process and may need readjustments and fine-tuning, based on run-time conditions



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.: Identication 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”

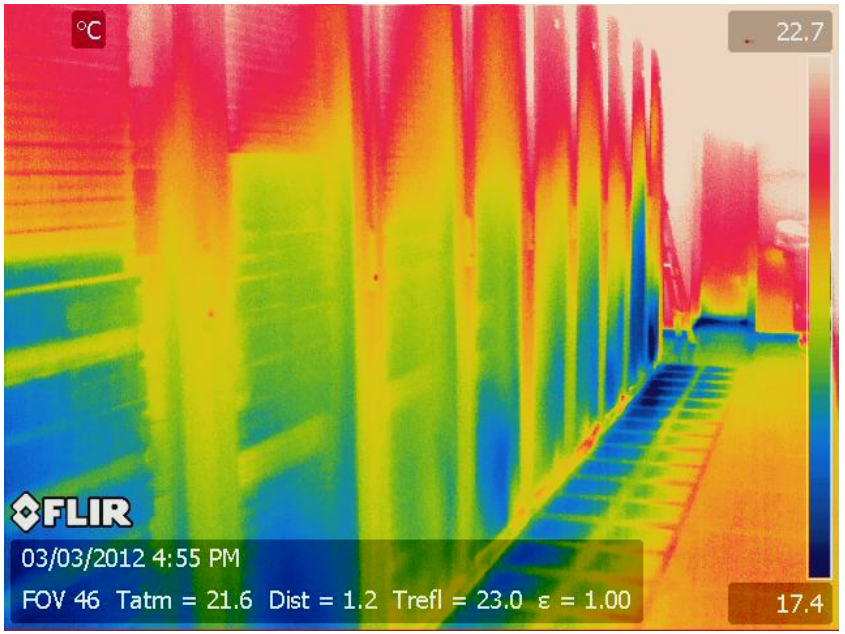
References

- [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?

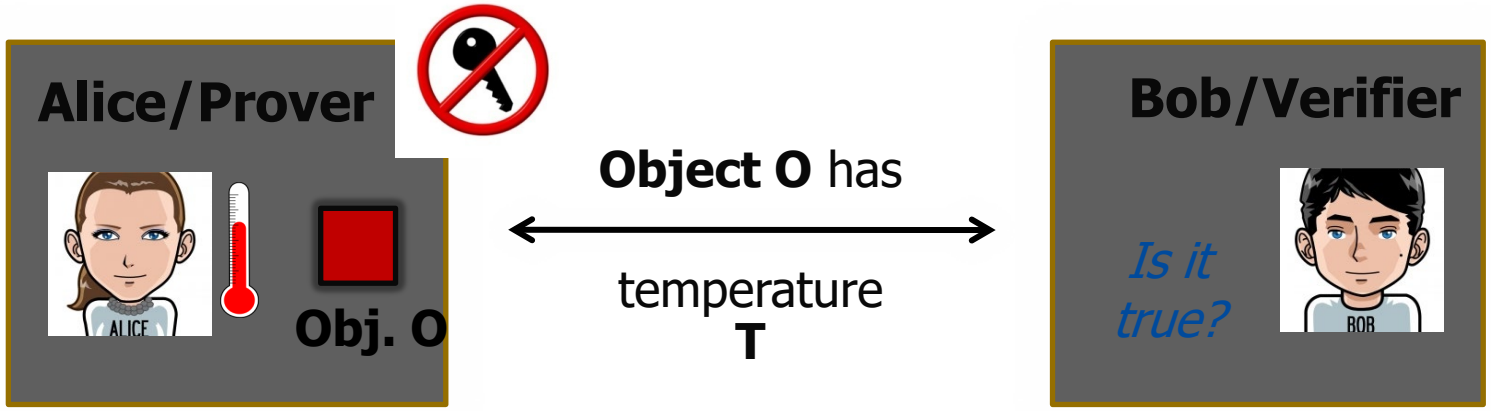
- Secure sensors are needed:
  - Operation safety, secrecy of sensitive data

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]

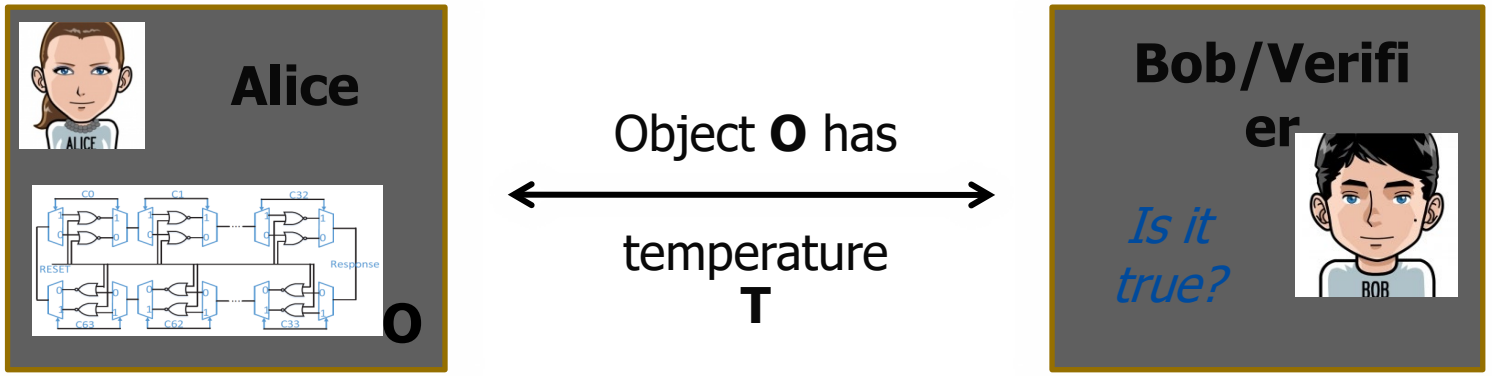
[1] Rivest, Ronald L. Illegitimi non carborundum. CRYPTO (2011).

Virtual Proofs of Temperature
(Or: Keyless Temperature Sensors)



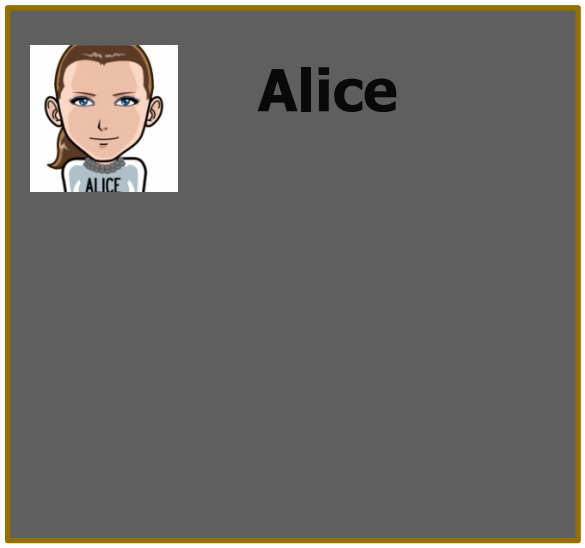
- Alice wants to prove to Bob that
  - a certain **object O**
  - is at temperature **T**
  - at the time of the execution of the VP
- **Classical approach:** Secure sensors with key known to Bob…
  - Buf, of course, we want no keys…

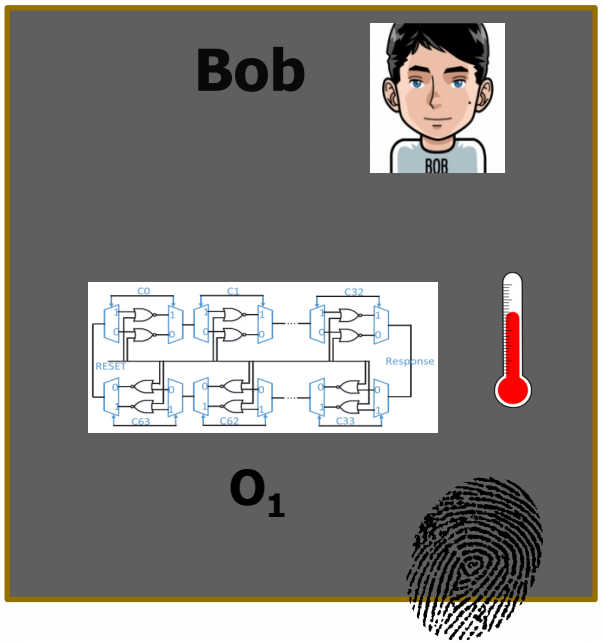
VPs of Temperature via Noise Sensitivity



- **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 $T_i \in \{T_1, \ldots, T_k\}$

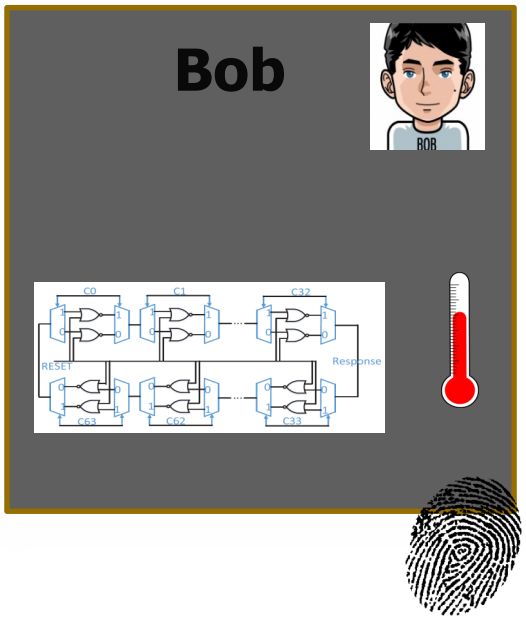




Private set-up phase:

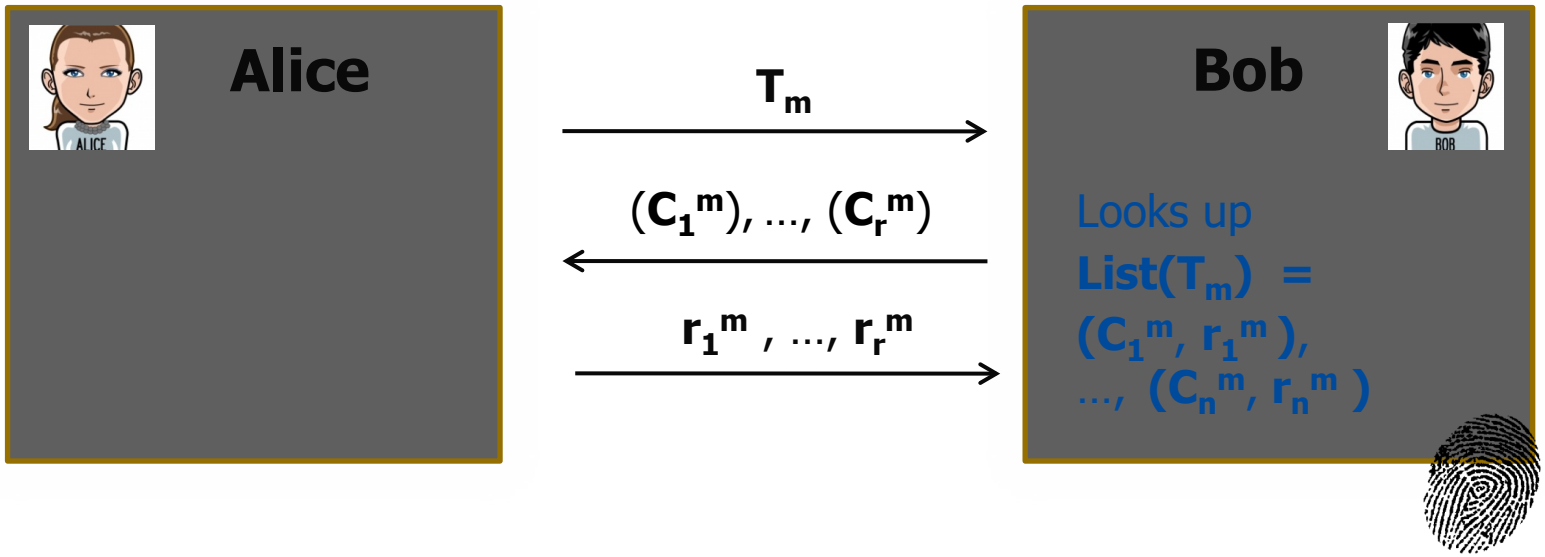
- For each temperature $\mathbf{T_i \in \{T_1, \ldots, T_k\}}$:
  - **Bob** puts the objects $\mathbf{O_1}$ at temperature $\mathbf{T_i}$
    Chooses **n** random **challenges** $C_1^i, \ldots, C_n^i$
    Measures the **n** resulting **responses**: $r_1^i, \ldots, r_n^i$
  - **Bob** creates&stores private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

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



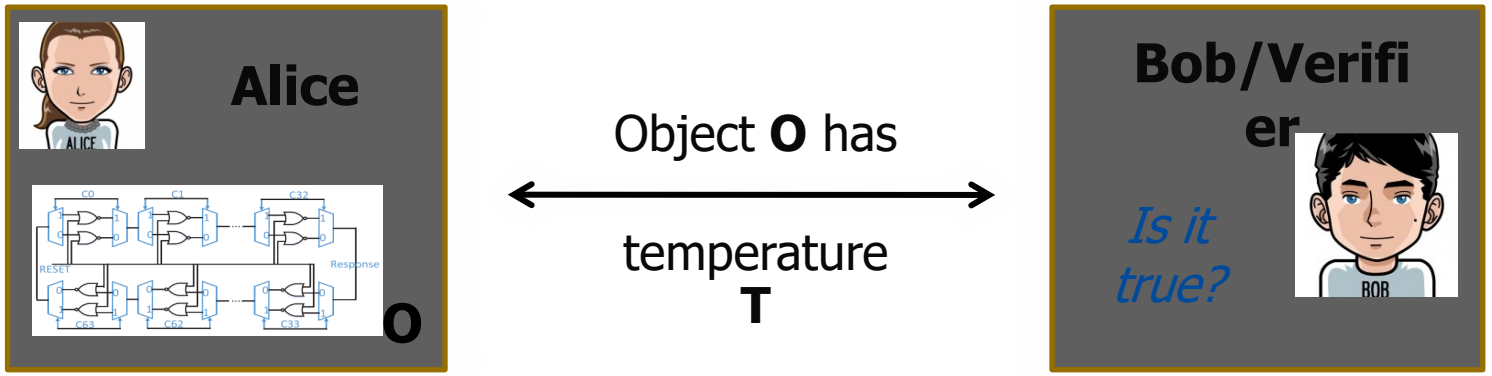
- Set-up phase closes,
  and the objects are transferred to **Alice**
- Some time may pass…
  (time for Alice to attack the system…)
- **VP starts!!**

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



- If pre-recorded values I_i^m in the **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