Max Austin Michael Clare

Grant Chris Will Jack Trey Yifan

07 True Random Number Generators (TRNGs)

Engr 399/599: Hardware Security
Andrew Lukefahr
Indiana University

Adapted from: Mark Tehranipoor of University of Florida

Course Website

engr599.github.io

Passcode:

Room

2-3-5

Write that down!

Project 1: Hardware Trojan

- Goal: "Corrupt" a working DES implementation

Shareable board in 14

- We give you DES in HW

- You need to:
  - Deploy DES
  - Corrupt DES

(remove JTAG)

Group Assignments

due next Wednesday

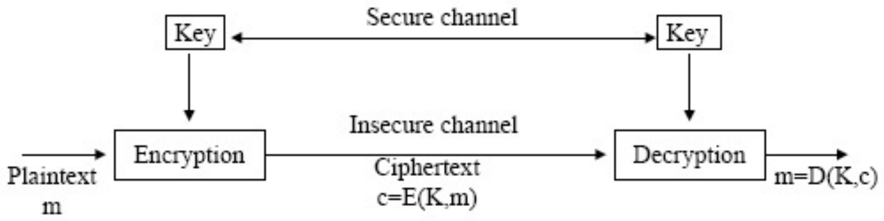
- Chris Sozio
- Will Fleming
- Clare Barnes

- Austin Parkes
- Max Harms
- Michael Foster

- Trey Peterson
- Yifan Zhang
- Jack Ruocco

Symmetric and Asymmetric Cryptosystems (1)

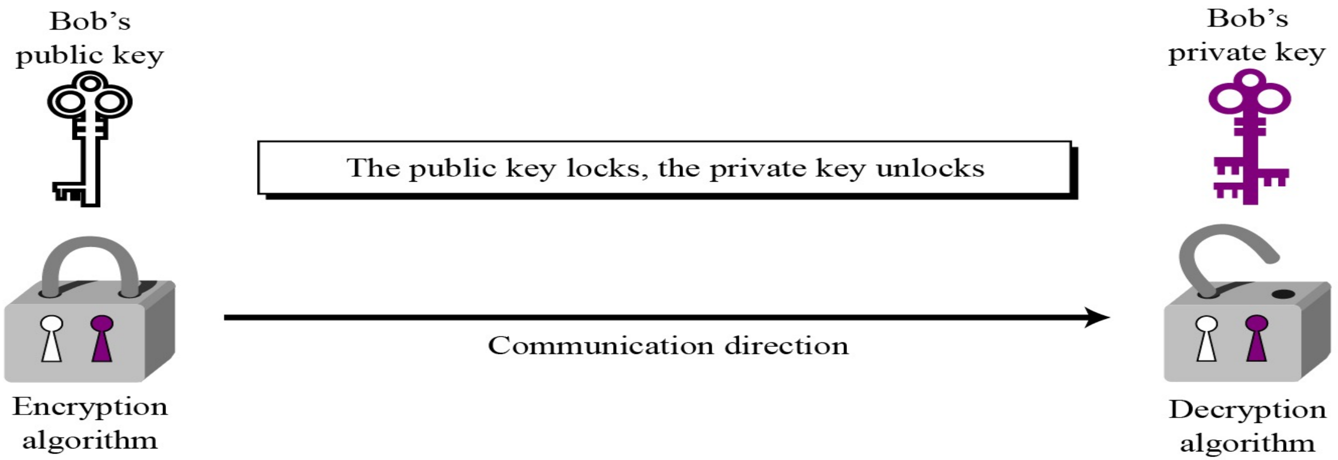
- Symmetric encryption = secret key encryption
  - $K_E = K_D$ — called a secret key or a private key
  - Only sender S and receiver R know the key

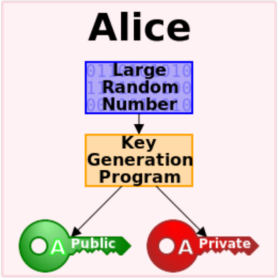


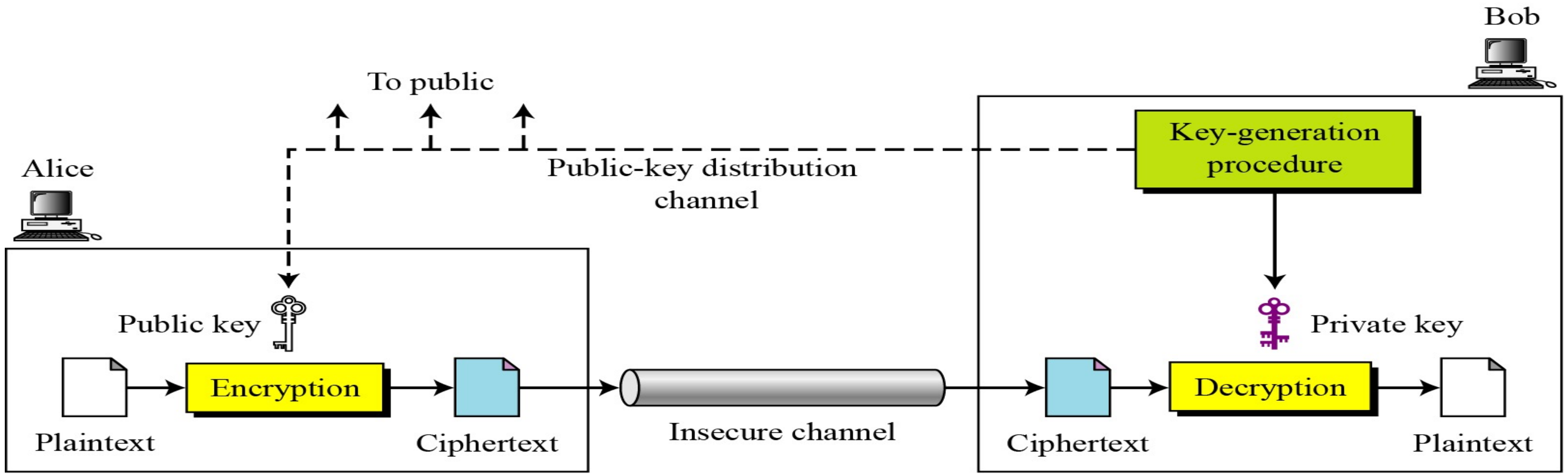
[cf. J. Leiwo]

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

Asymmetric Security







Attacks

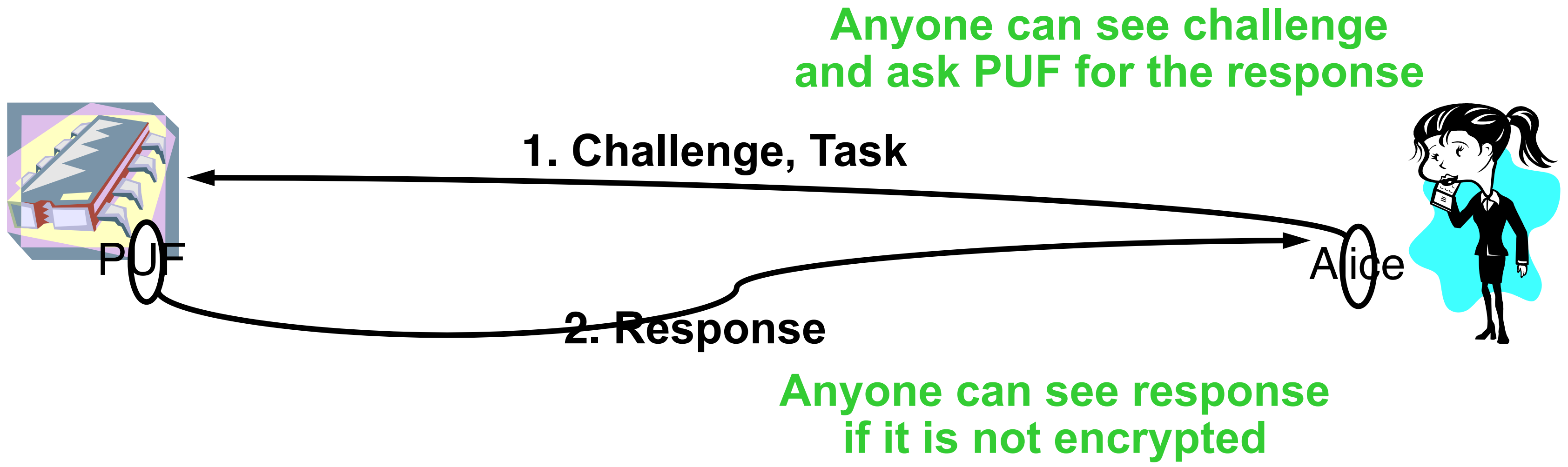
Software-only protection is not enough. Non-volatile memory technologies are vulnerable to invasive attack as secrets always exist in digital form

Sharing a Secret with a Silicon PUF

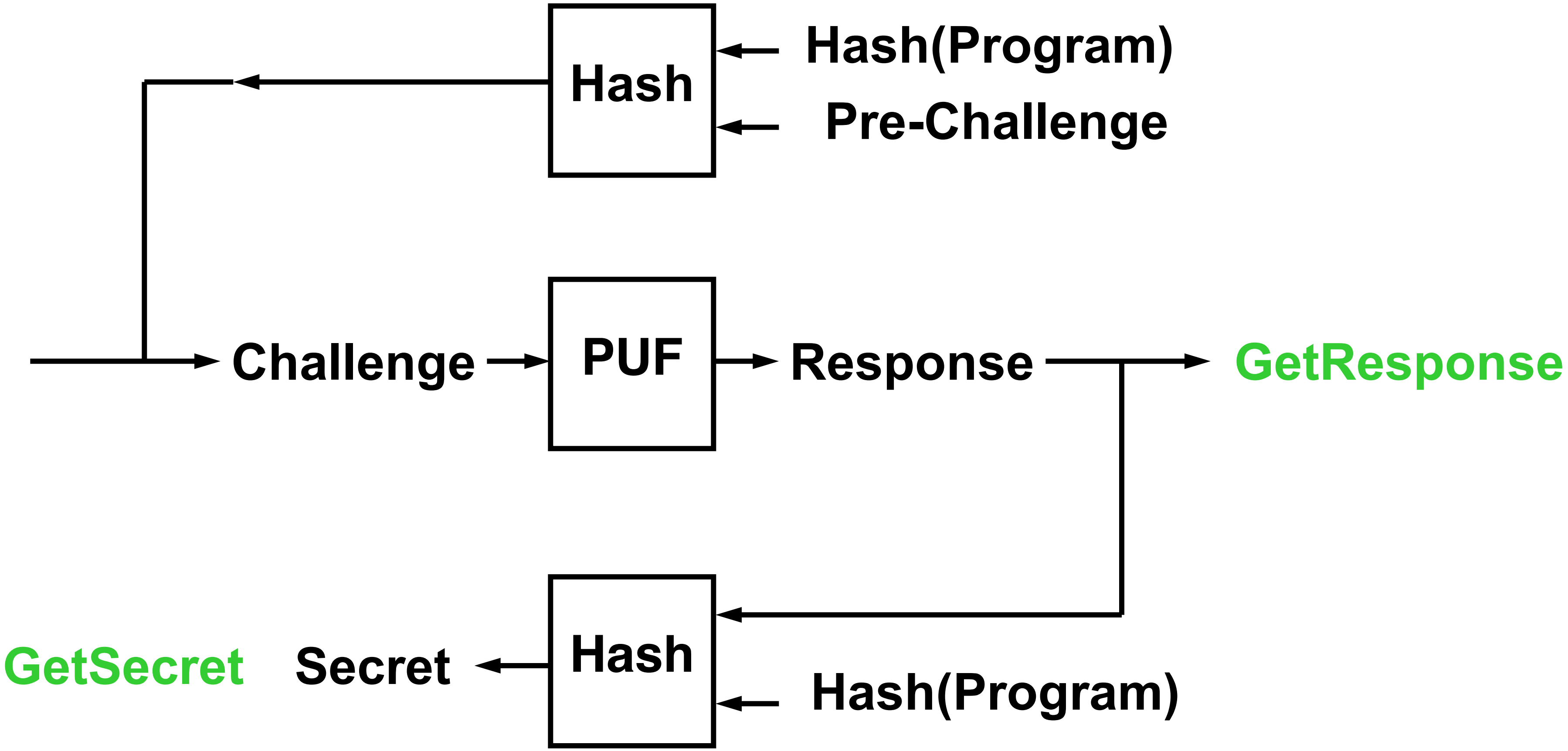
Suppose Alice wishes to share a secret with the silicon PUF

She has a challenge response pair that no one else knows, which can authenticate the PUF

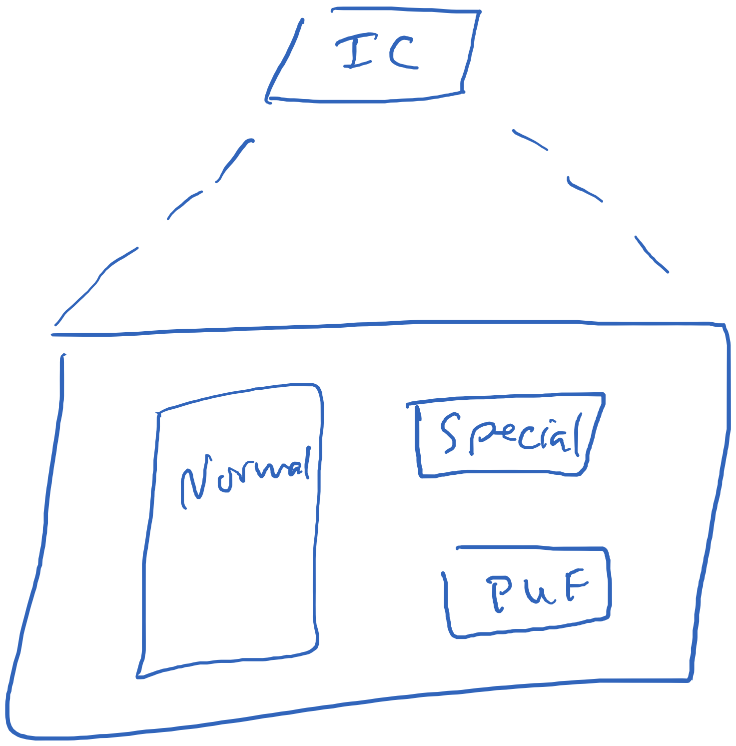
She asks the PUF for the response to a challenge



Controlled PUF Implementation



~~Software~~ Licensing

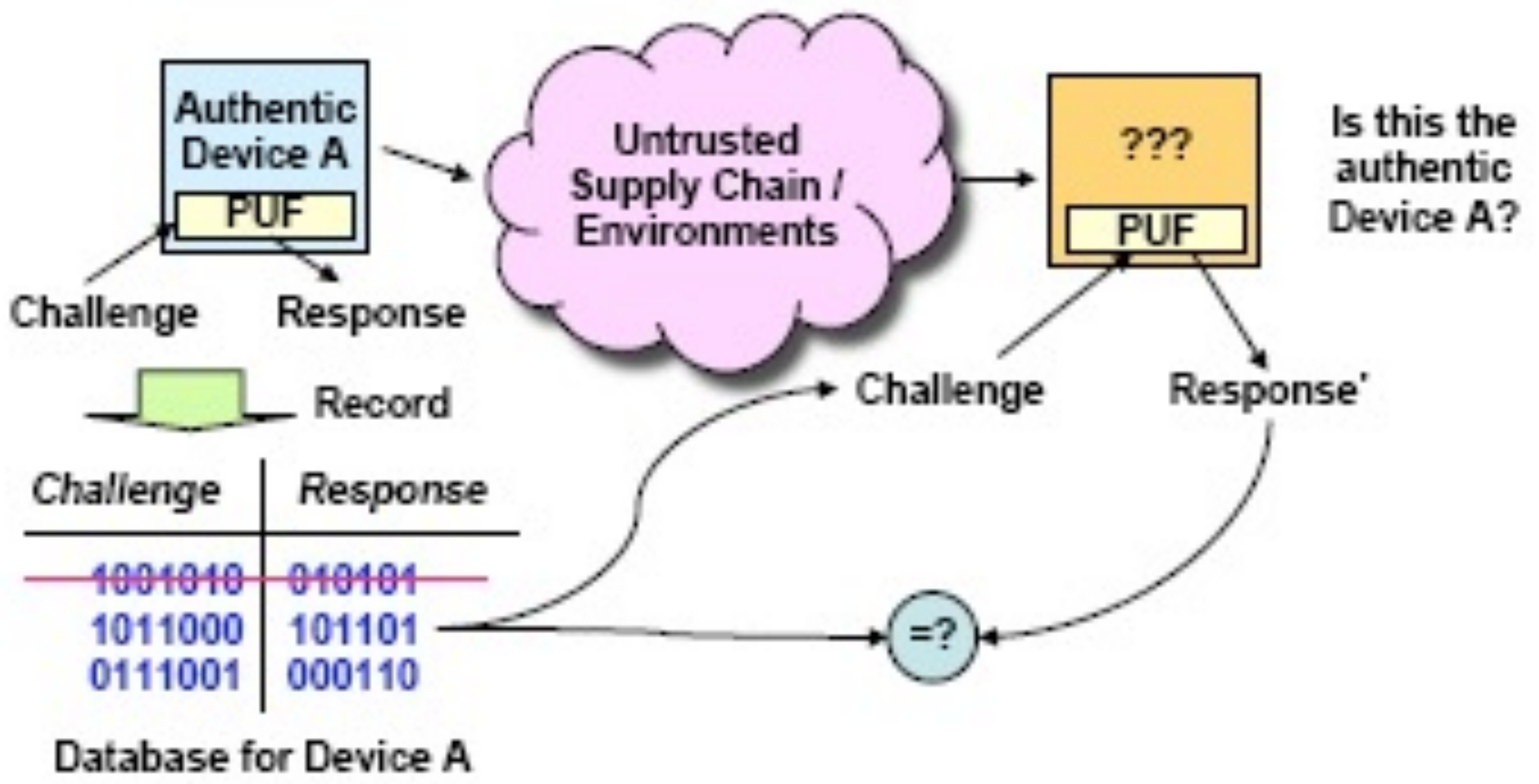


Challenge → Response

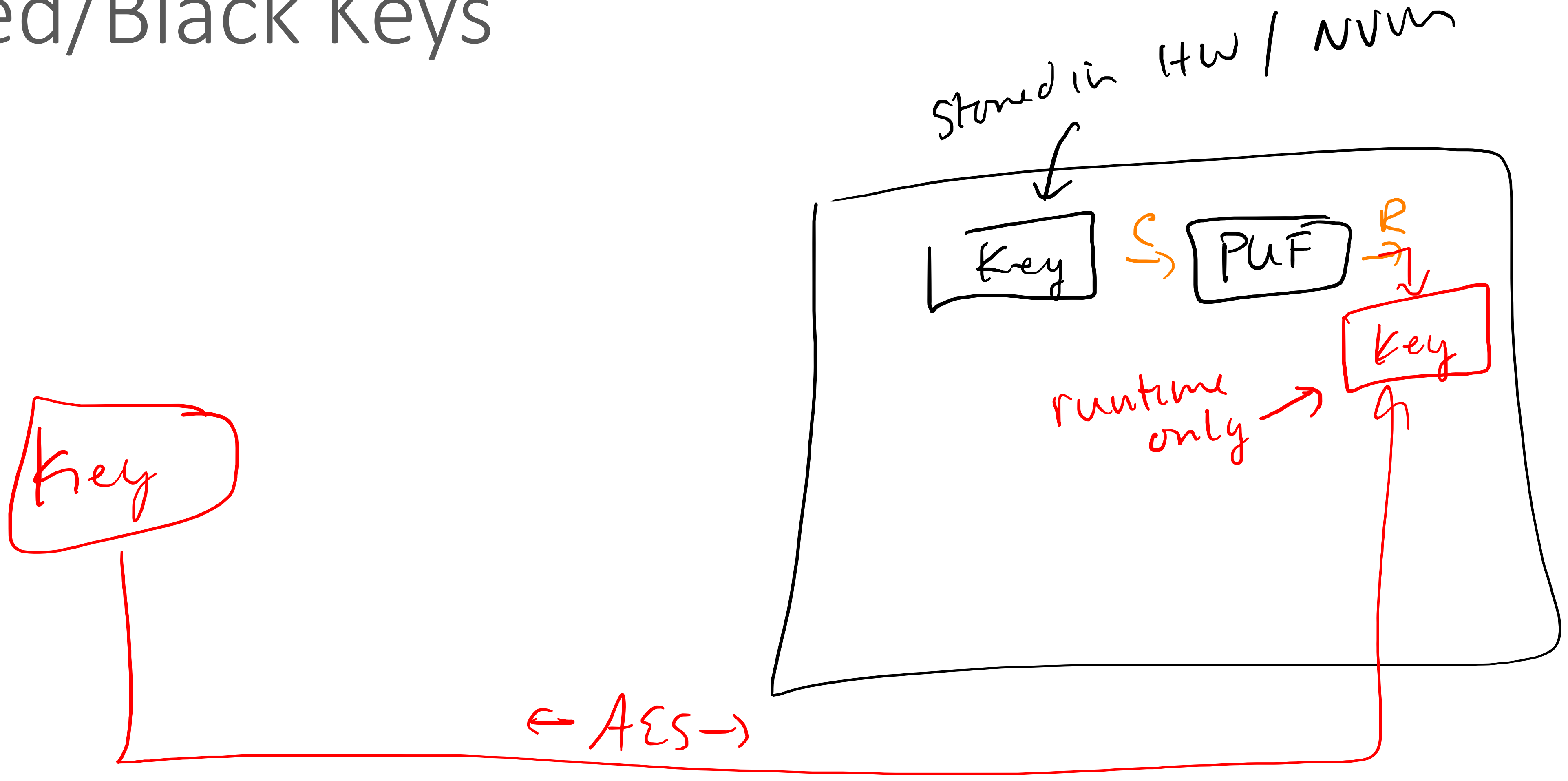
Color → flower

Applications – Authentication

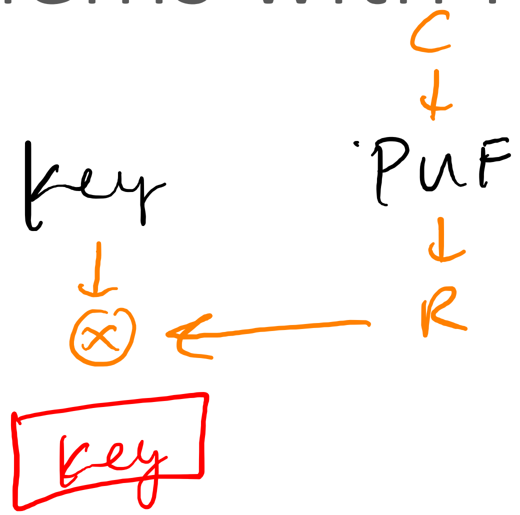
- Same challenges should not be used to prevent the man-in-the-middle attacks

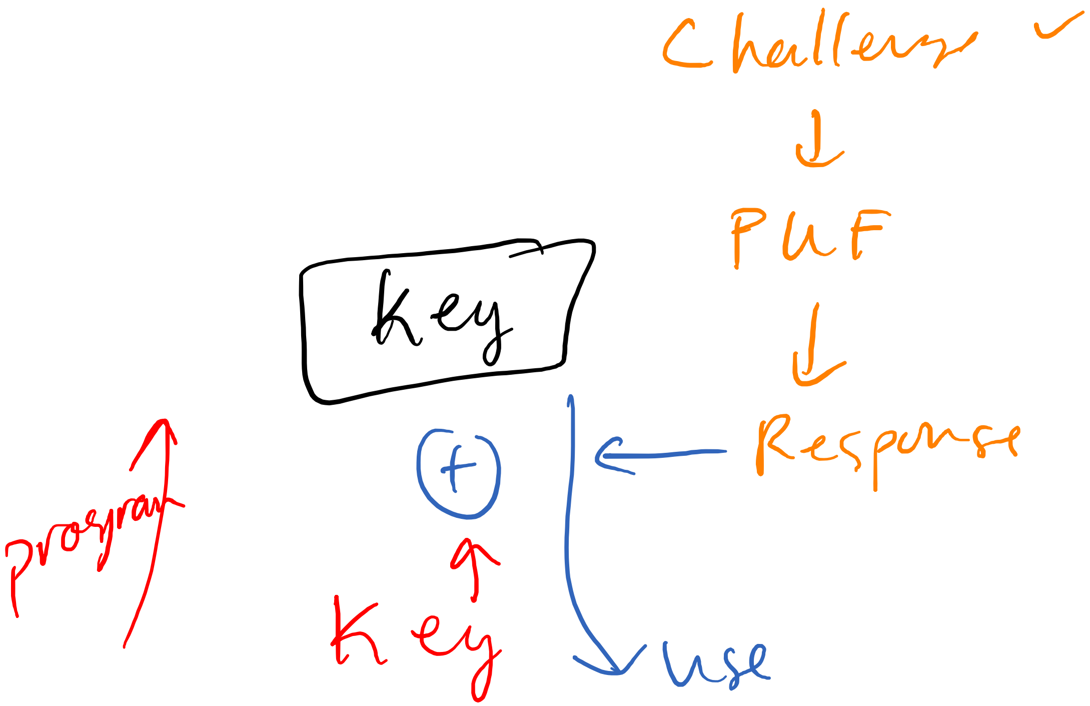


Red/Black Keys



Problems with PUFs





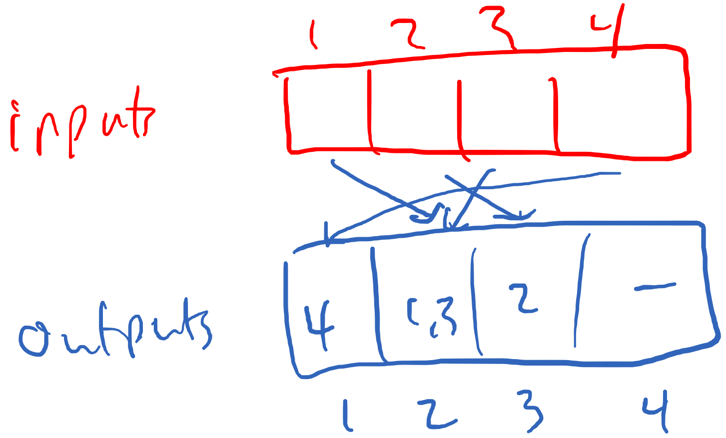
Puf Problems

challenge → response

Color → "Strawberry"

ice-cream → strawberry

fruit → strawberry



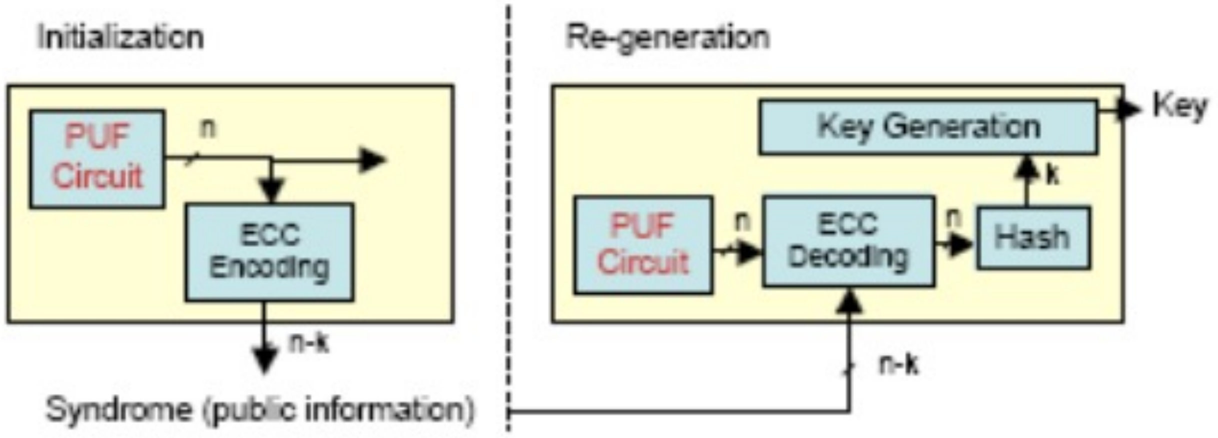
→ similar answers across pufs

→ silimilar answers w/in puf

→ dissimilar responses w/in PUF

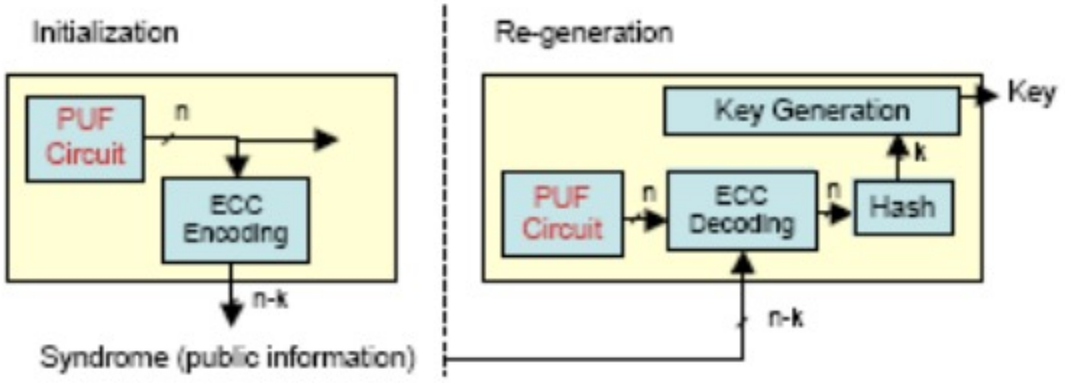
Application – Cryptographic Key Generation

- The unstability 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



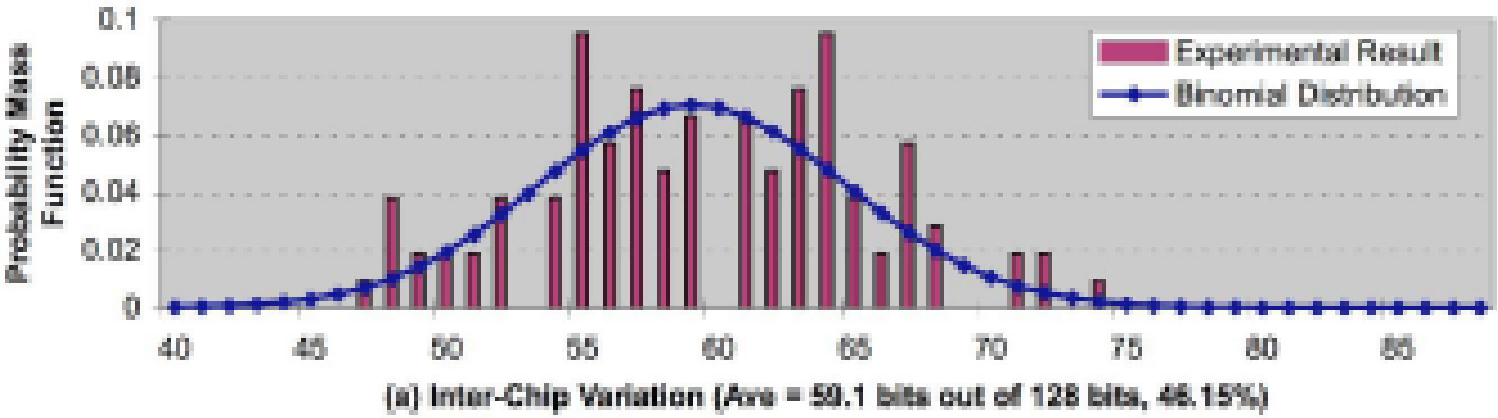
Crypto Key Generation

- Initialization: a PUF output is generated and error correcting code (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



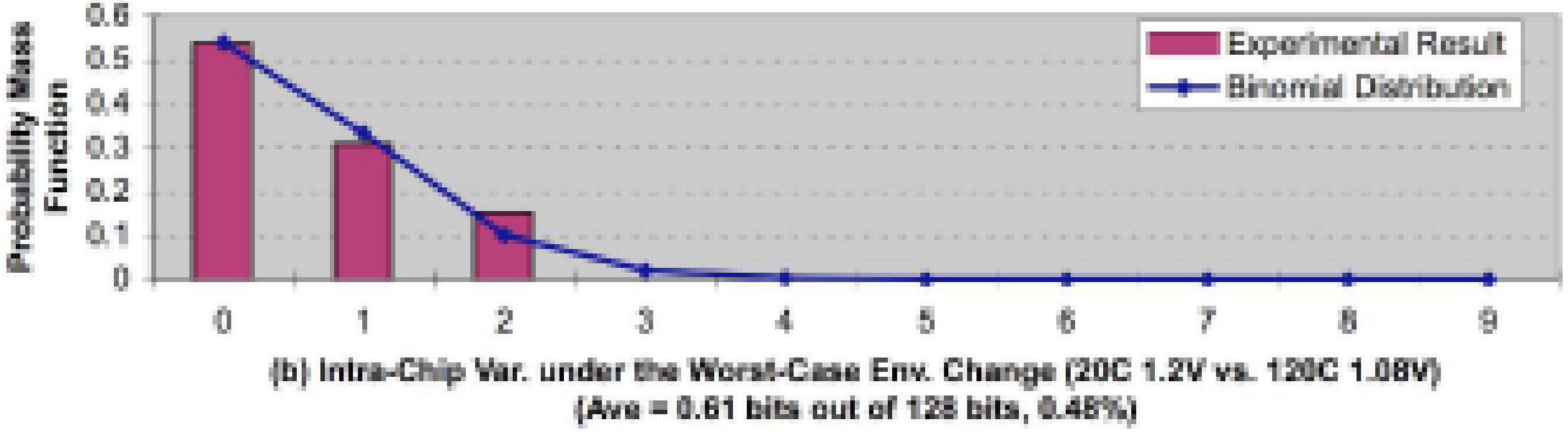
The Probability Distribution for Inter-chip Variations

- 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



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

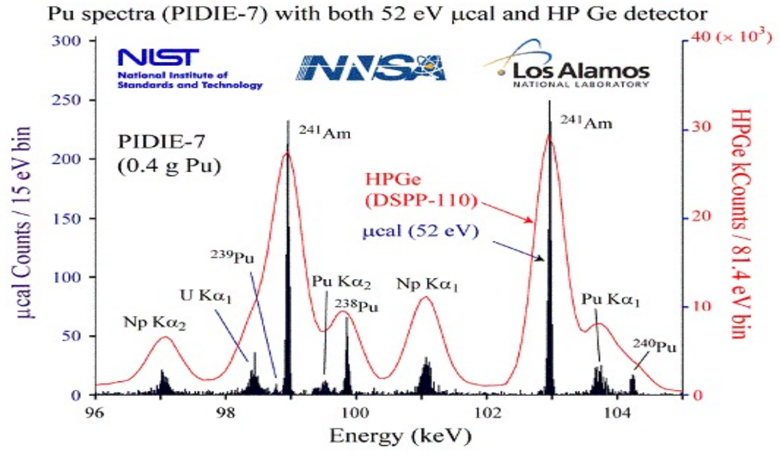
- **Pseudo-random number generator:**
  - A polynomial-time computable function f (x) that expands a short random string x into a long string f (x) that appears random

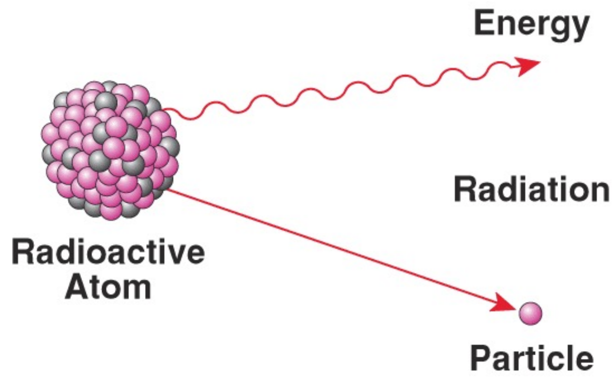
- **Not truly random in that:**
  - Deterministic algorithm
  - Dependent on initial values (seed)

- **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.

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.

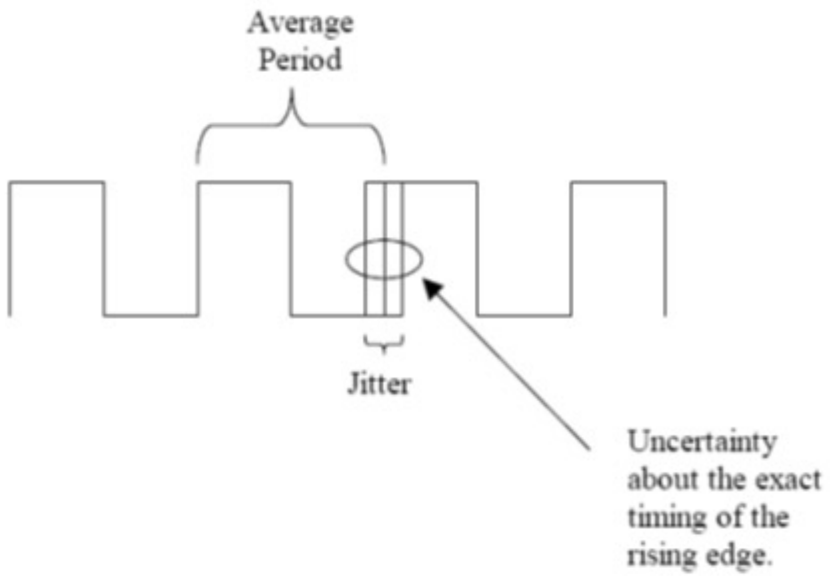
Set of Requirements

- 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.

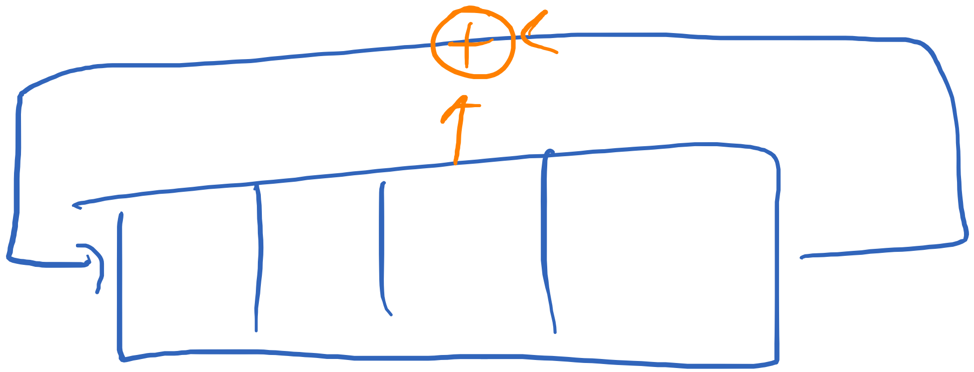
Method : Clock Jitter

- 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



Linear Feedback Shift Register



1 2 3 4 ← original

4 1 2 3 ← shift

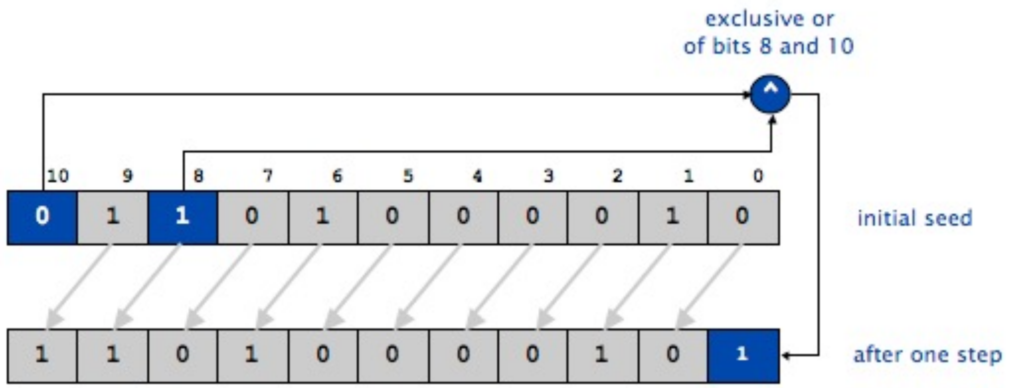
$4 \oplus 3$ 1 2 3

7 1 2 3

↙ linear feedback shift

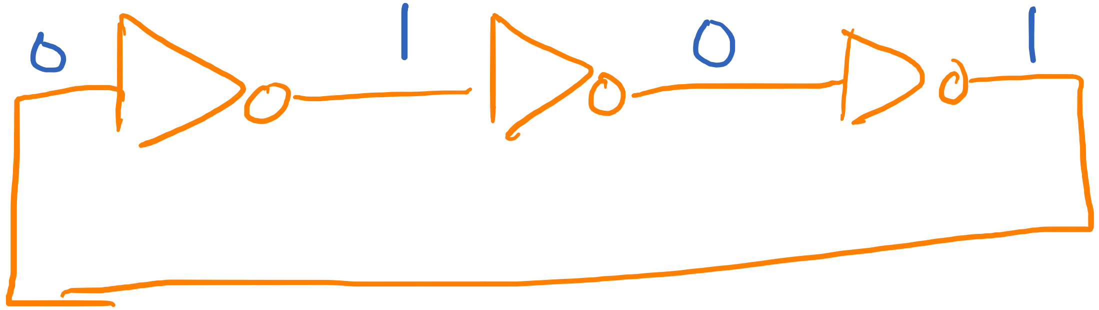
$$\begin{array}{r} 100 \\ \oplus\ 011 \\ \hline 111 \end{array}$$

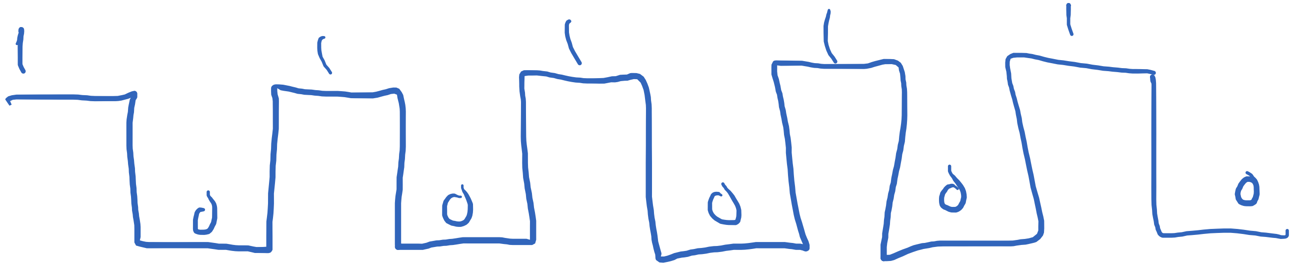
Linear Feedback Shift Register



One step of an 11-bit LFSR with initial seed 01101000010 and tap at position 8

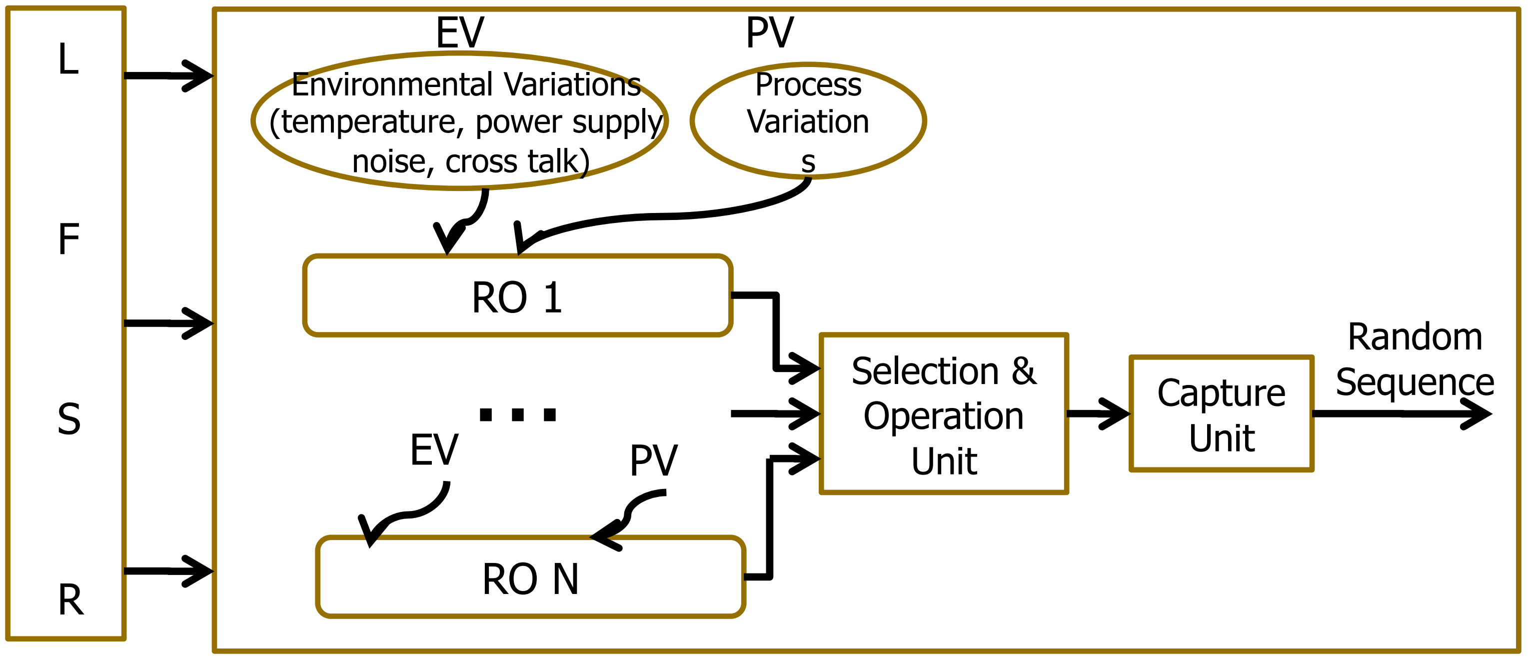
Ring Oscilator





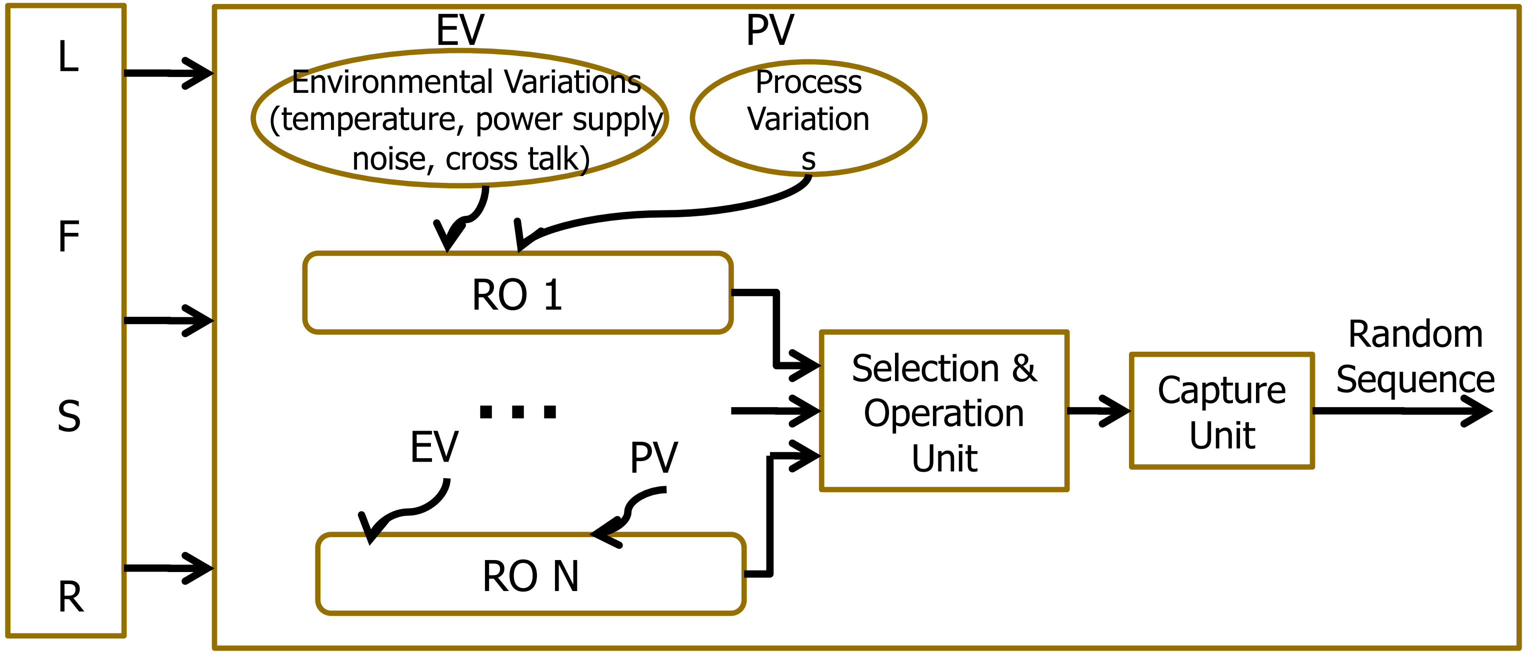
TRNG Structure

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



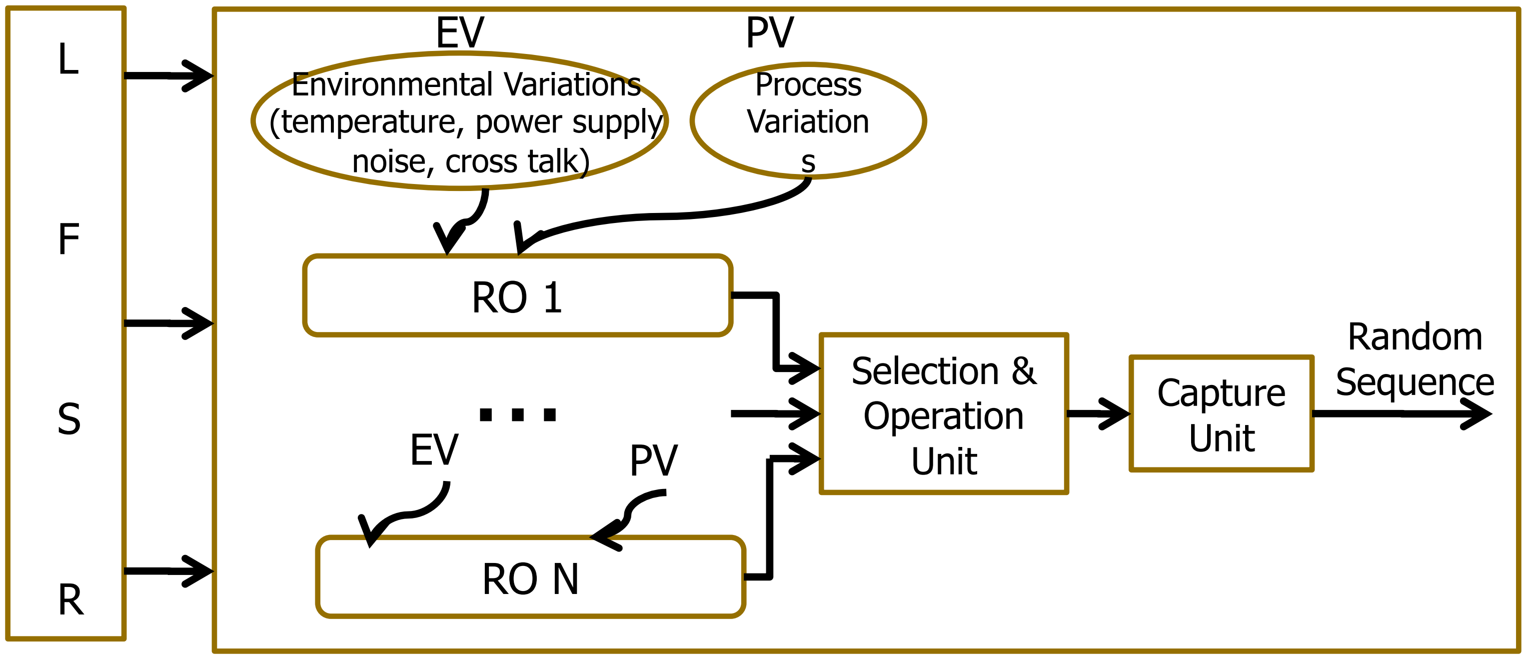
TRNG Structure

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



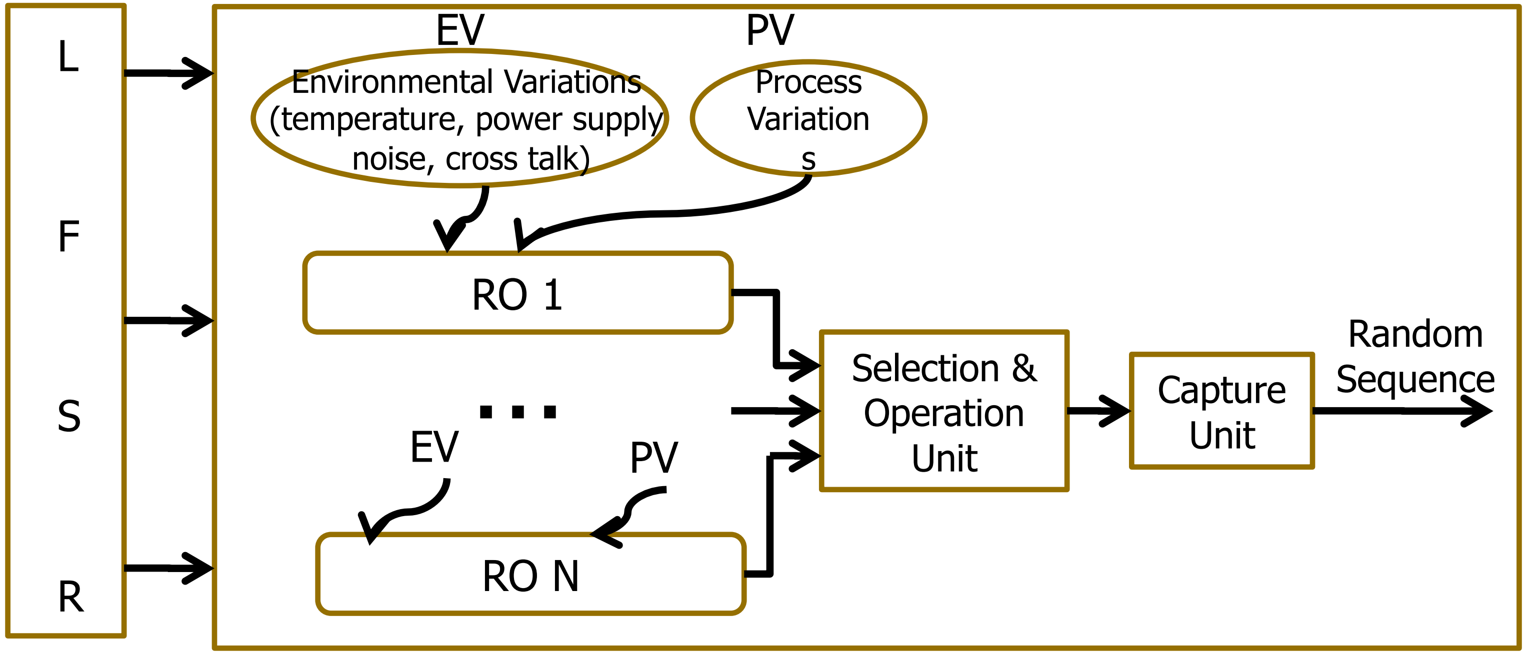
TRNG Structure

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

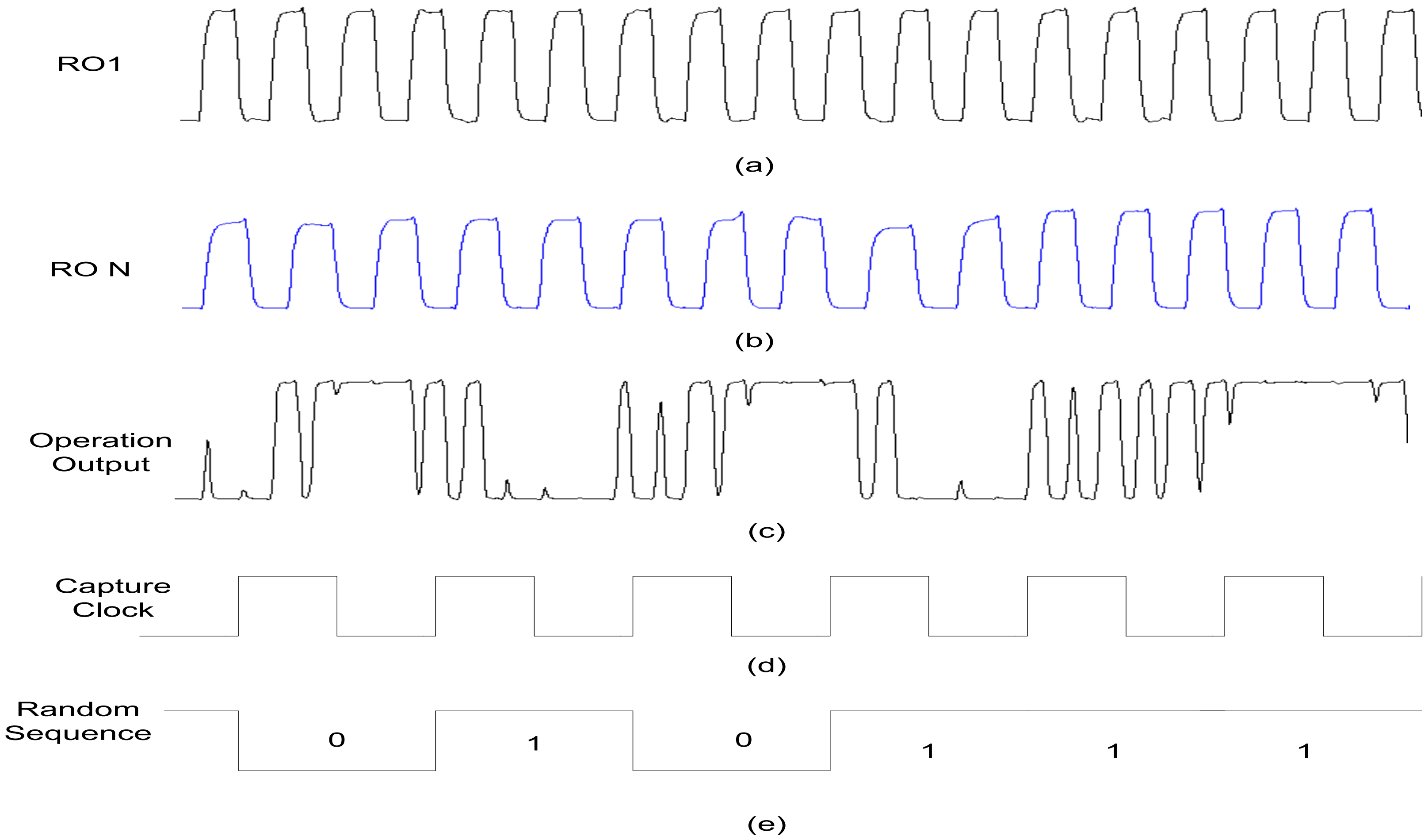


TRNG Structure

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



TRNG Output


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

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.