# PUFs FOR FPGA SECURITY

By Ankith Srinivasan

## Title of Contents

## Introduction to PUFs

- A Physical Unclonable Function (PUF) is a hardware security primitive that generates a unique digital fingerprint for each FPGA.
- PUFs are based on inherent manufacturing variations in each chip, making them nearly impossible to clone or replicate.
- The use of PUFs in FPGA security is an effective way to enhance the protection of intellectual property and sensitive data.
- PUFs serve as a critical security feature by creating a hardware-based fingerprint that can be used for device authentication and encryption key generation.
- This inherent uniqueness makes it extremely difficult for adversaries to clone or counterfeit FPGAs.
- PUFs add an additional layer of security, reducing the risks associated with FPGA-based applications.

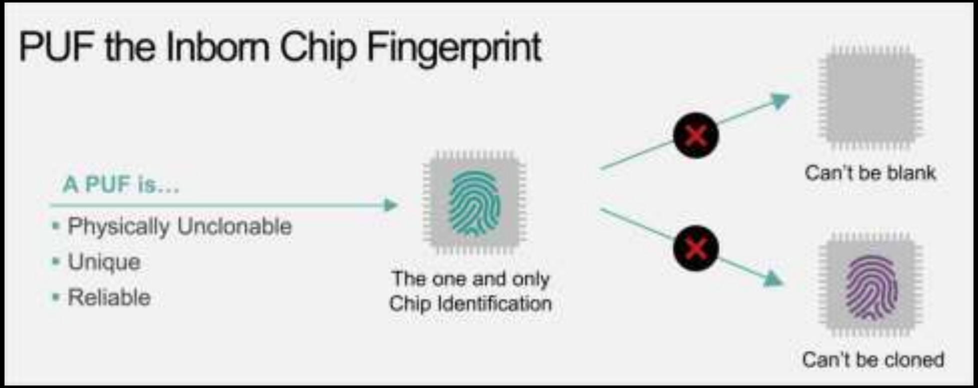


Image from: PUF

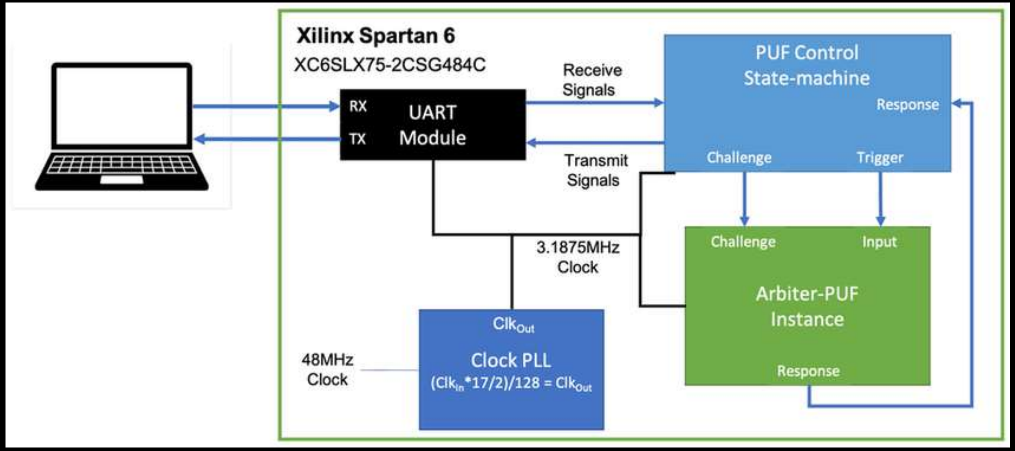
## FPGA Security Challenges

- FPGAs are vulnerable to various attacks, including reverse engineering, IP theft, and tampering, due to their reprogrammable nature.
- Protecting FPGA designs is essential, especially in applications where security is critical, such as in aerospace, defense, and IoT.
- PUFs offer a robust solution to these challenges by providing a unique hardware-based identity for each FPGA.

Image form: FPGA

## How exactly does a PUF work?

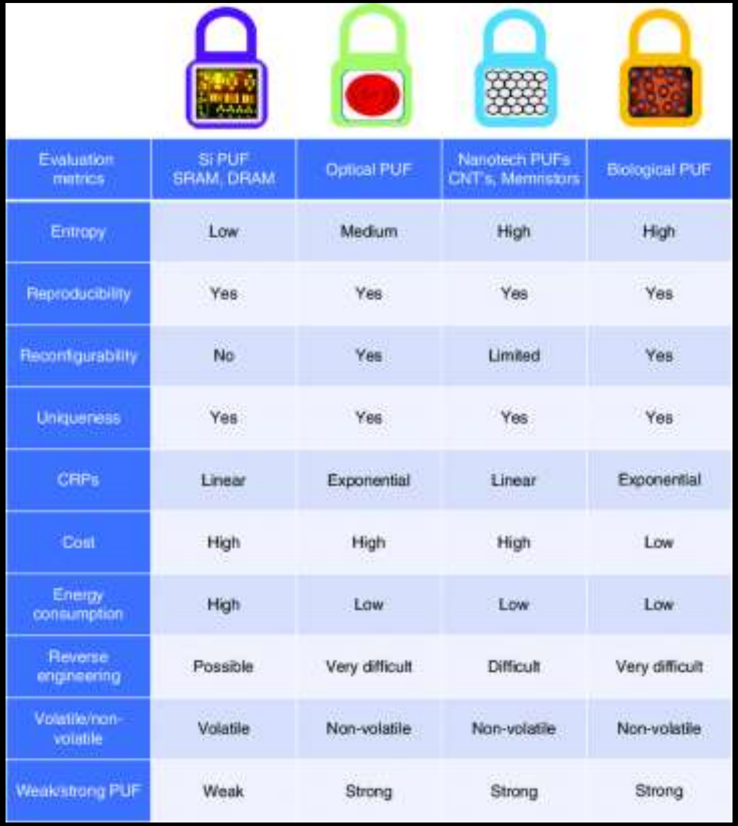
- A PUF typically operates on a challenge-response mechanism. A challenge is a specific input provided to the PUF, often in the form of a digital signal or command.
- When the challenge is applied to the PUF, the physical variations in the electronic components lead to unique responses. These responses are generated based on the specific hardware characteristics of the device.
- The keys are not physically stored anywhere.
- The critical property of a PUF is its uniqueness. The response produced by one device when given a particular challenge is different from the responses of other devices, even if they are manufactured from the same production process.
- While the responses are unique, they are also repeatable.
- PUFs can be classified into 2 types strong and weak.
- PUFs are low cost and consume very little power.



Ref:image

## Types of PUFs

- Several types of PUFs are employed in FPGA security. These include:
  - DRAM PUF: leverages the inherent variations in DRAM cells to generate a unique fingerprint for electronic devices.
  - Arbiter PUF: Relies on the delay differences in paths through a set of logic gates.
  - SRAM PUF: Utilizes the startup state of SRAM cells to generate a unique key.
  - Ring Oscillator PUF (RO-PUF): Exploits variations in the frequency of ring oscillators.
  - Delay PUF(D-PUF): A Delay PUF is based on the differences in propagation delays between signals through similar paths in the circuit.
  - Thermal noise PUF: Thermal Noise PUFs leverage the random fluctuations in electronic components caused by thermal effects.
- The choice of PUF type depends on the specific security requirements and FPGA architecture.



| Evaluation metrics | Si PUF SRAM, DRAM | Optical PUF | Nanotech PUFs CNT's, Memristors | Biological PUF |
|---|---|---|---|---|
| Entropy | Low | Medium | High | High |
| Reproducibility | Yes | Yes | Yes | Yes |
| Reconfigurability | No | Yes | Limited | Yes |
| Uniqueness | Yes | Yes | Yes | Yes |
| CRPs | Linear | Exponential | Linear | Exponential |
| Cost | High | High | High | Low |
| Energy consumption | High | Low | Low | Low |
| Reverse engineering | Possible | Very difficult | Difficult | Very difficult |
| Volatile/non-volatile | Volatile | Non-volatile | Non-volatile | Non-volatile |
| Weak/strong PUF | Weak | Strong | Strong | Strong |

Ref:image

## DRAM PUF

- A standard DRAM cell works with a single capacitor to hold a stored charge as a binary state, and a pass transistor that controls the flow of charge to and from the capacitor.
- Due to device non-idealities, such as leakage, the charge on the capacitor tends to leak over time which causes the cell to lose state.
- Importantly for the purposes of a PUF, each individual DRAM cell leakage rate is highly affected by process variations in the cell's manufacturing.
- To counteract this, all DRAM cells perform periodic refresh commands which reassert charge to "refresh" the storage capacitor.
- A DRAM PUF, on the other hand, works by pausing this refresh for a longer-than-usual specified interval of time and seeing how the cells have changed state due to leakage.
- In this instance, the "challenge" is the original binary value asserted to an array of DRAM cells, and the response is the value of that array after the given time interval.
- This technique may be used to generate truly random numbers for use in a cryptographic key generation, or it may be used for device identification for counterfeit protection.

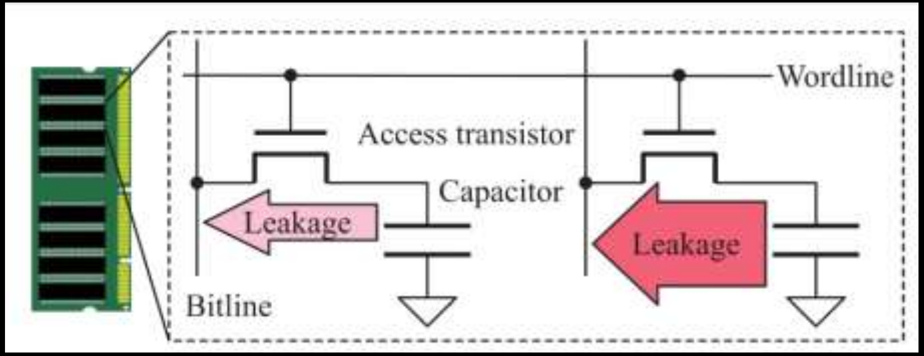
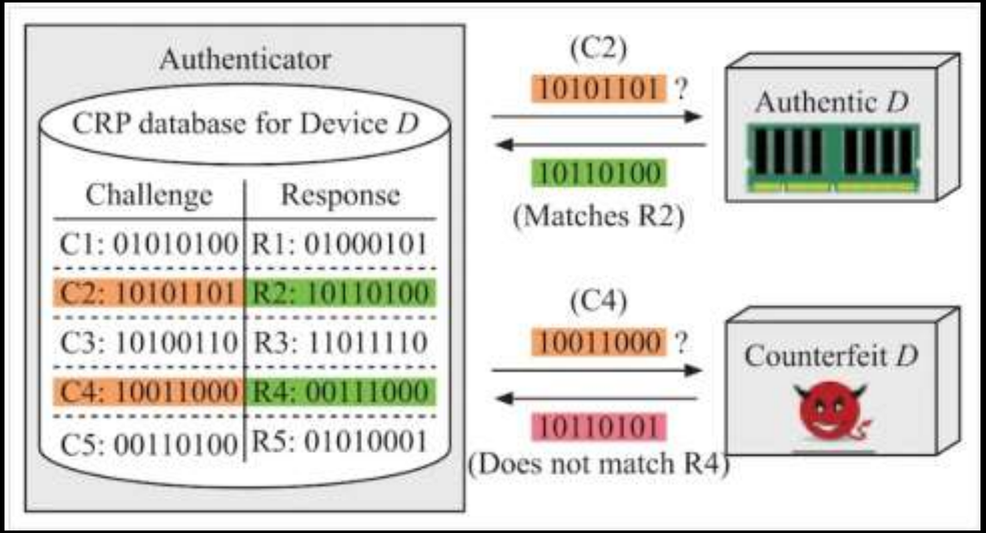


Image from: DRAMPUF

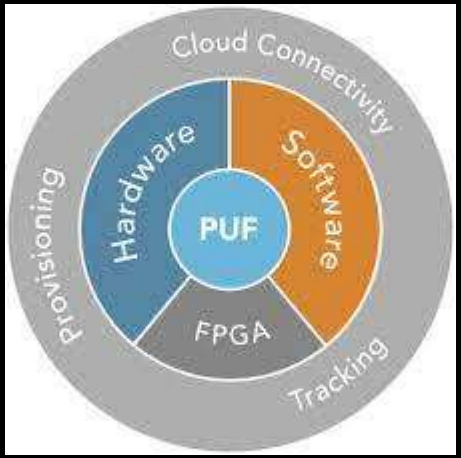


| Challenge | Response |
|---|---|
| C1: 01010100 | R1: 01000101 |
| C2: 10101101 | R2: 10110100 |
| C3: 10100110 | R3: 11011110 |
| C4: 10011000 | R4: 00111000 |
| C5: 00110100 | R5: 01010001 |

Image from: DRAMPUF

# How to implement PUFs on FPGA

- Choose the specific type of PUF you want to implement based on your security requirements and the FPGA's architecture.
- Prepare your FPGA board and development environment for PUF implementation. This may involve ensuring the FPGA is properly powered and configured.
- Create or obtain the PUF design, which typically includes the PUF core logic. Many FPGA manufacturers provide IP cores or libraries for PUFs that you can use.
- Use hardware description languages (HDL) like VHDL or Verilog to describe how the PUF is integrated into the FPGA's logic.
- Implement the logic that captures the response generated by the PUF when presented with a challenge. The response may need to be processed and filtered as necessary.
- Decide how to handle and store the challenge-response pairs. This could involve storing the data in memory or interfacing with other components in your system.
- Test the PUF implementation to ensure that it produces unique, reliable, and secure responses.
- Implement additional security measures to protect the PUF and its responses.
- Validate the entire system to ensure it meets your security objectives and is robust against known threats.



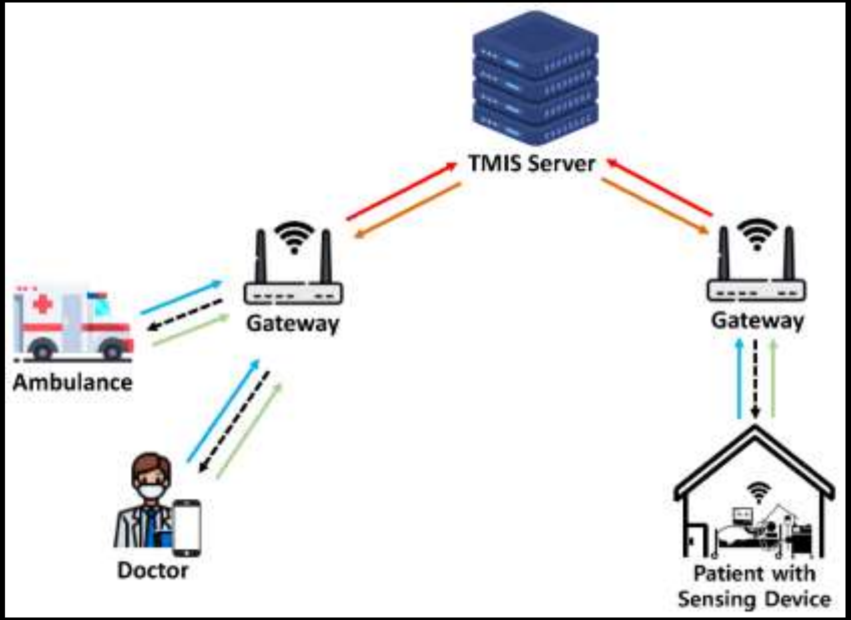
Ref:image

## Challenges

Ref:image

Ref:image

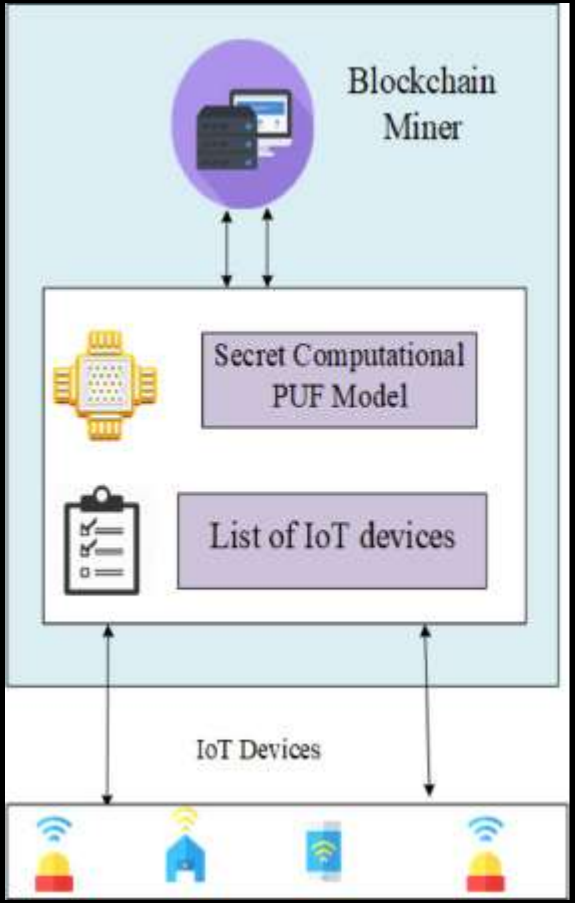
- Despite their effectiveness, PUFs have some challenges, such as reliability issues and environmental variations.
- The server containing the CRPs can get hacked.
- Ongoing research is focused on improving the reliability and resilience of PUFs.
- PUFs may experience aging effects, potentially leading to changes in their responses and impacting reliability.
- External noise and disturbances can influence PUF behavior, raising concerns about reliability in noisy operational environments.



Ref:image

## Mitigation Strategies

- **Error Correction Techniques:** Implementing error correction codes to enhance the reliability of PUF responses and mitigate the impact of variations.
- **Temperature and Voltage Stabilization:** Designing FPGAs with measures to stabilize temperature and voltage, reducing the impact of environmental variations.
- **Dynamic Configuration Updates:** Periodic updates to PUF configurations can help adapt to aging effects and maintain reliability over the FPGA's lifecycle.
- Be careful how you implement the CRPs.
- **Future directions:** may involve combining PUFs with other security techniques for enhanced protection.



Ref:image

## Conclusion

- *PUFs in FPGAs:* Despite challenges, PUFs remain a valuable tool for enhancing security and reliability in FPGA-based systems.
- *Ongoing Research:* Continuous research and development are crucial to address reliability concerns and unlock the full potential of PUF technology.

Ref:image

## References

- Paper: https://www.iosrjournals.org/iosr-jvlsi/papers/vol4-issue1/Version-1/D04111621.pdf
- Article: https://www.frontiersin.org/articles/10.3389/fsens.2021.751748/full

THANK YOU