

Educational content for PUF and PUF based computation

Amir Ali-pour

Université Grenoble Alpes (UGA)

EMNESS

Outline

1. Introduction to PUF
 - What is PUF
 - What are fields of utilizations of PUF in general
 - Physical characteristic of PUF
 - Commercial PUFs
2. PUF Categories and features
 - Strong PUF
 - Structures
 - How it works (homework)
 - Weak PUF
 - Structures
 - How it works (homework)
 - Features of a good PUF
 - Entropy, Reliability, Randomness, diffuseness, uniqueness

Assignment_1:

- Simulate a simple Arbiter PUF using Python/MATLAB

3. PUF based protocols
 - Authentication
 - How it works (homework)
 - Key generation
 - How it works (homework)
 - Random Number generation
 - How it works (homework)

Assignment_2:

- Implement an authentication protocol using the PUF simulation

4. Issues and challenges with PUF
 - PUF model-building attacks
 - How it works (homework)
 - PUF side channel attacks
 - PUF Fault injection attacks
 - Replay Attack

Assignment_3 (optional):

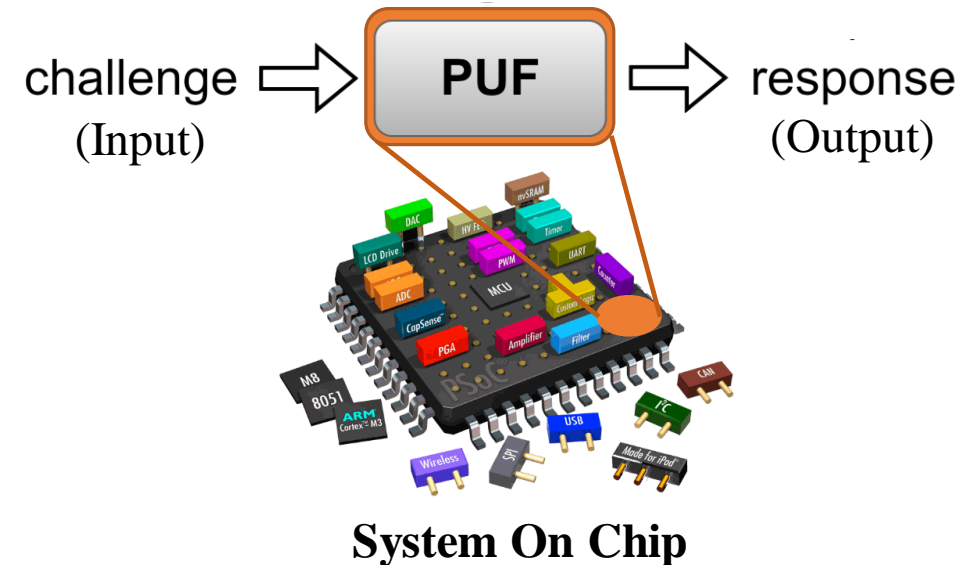
- Model Arbiter PUF using Pytorch or Scikit-learn (python)

5. Existing solutions
6. Emerging topics of and around PUF

Project

Ch1: Introduction: What is PUF

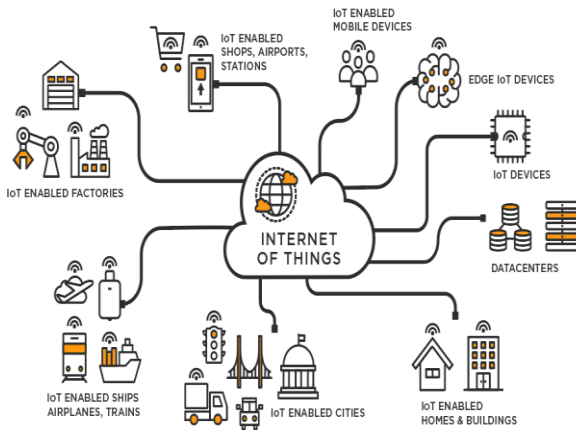
- Physically Unclonable Function:
 - Is embodied in the physical characteristic of silicon devices.
 - Is represented as a randomized Fingerprint generator
 - PUF Fingerprint(s) are unique to each chip
- Key points:
 - ❖ Needs no NV Memory to store secret values
 - ❖ PUF physically is more tamper resistant than NV memories. Since it is related to micro process variations, if we tamper the PUF, we destroy the micro features.



Ch1: Introduction: Fields of Utilization

IoT

- IoT edge device authentication.
- Providing core values for lightweight device-to-device communication encryption.



<https://www.tibco.com/reference-center/what-is-the-internet-of-things-iot>

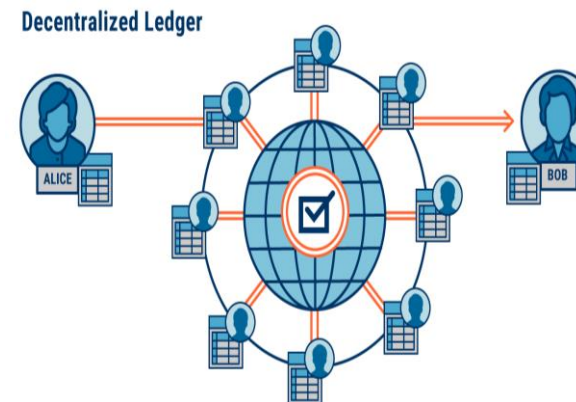
Smart Card

- Lightweight implementation that provides abundance of fingerprints for authentication.
- Encryption should be strong but can be possible including some post-processing mechanisms.



Block Chain

- The very large CRP space of PUF is the major benefit here.



<https://www.cbinsights.com/research/what-is-blockchain-technology/>

Smart Vehicles

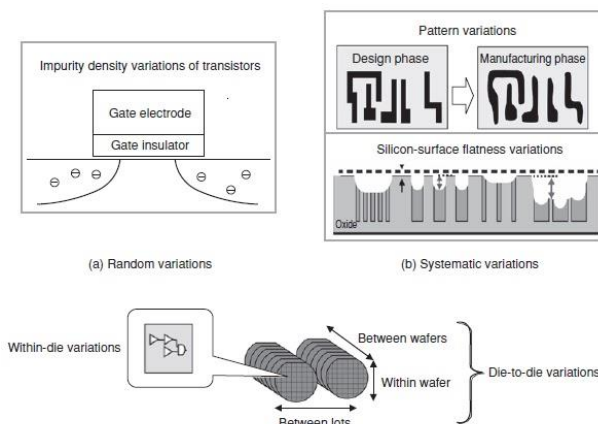
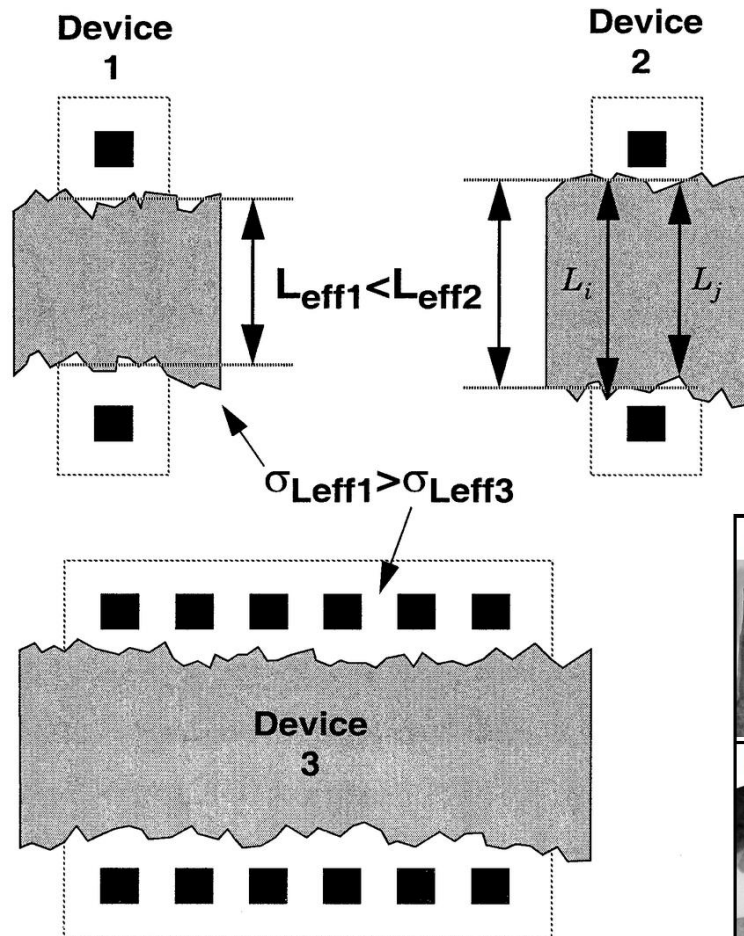
- Can serve as a reliable and secure primitive for authentication.



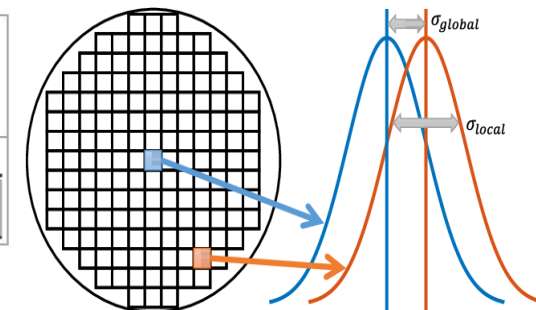
<https://www.digi.com/blog/post/what-is-connected-vehicle-technology-and-use-cases>

Ch1: Introduction: Physical characteristic & process variation

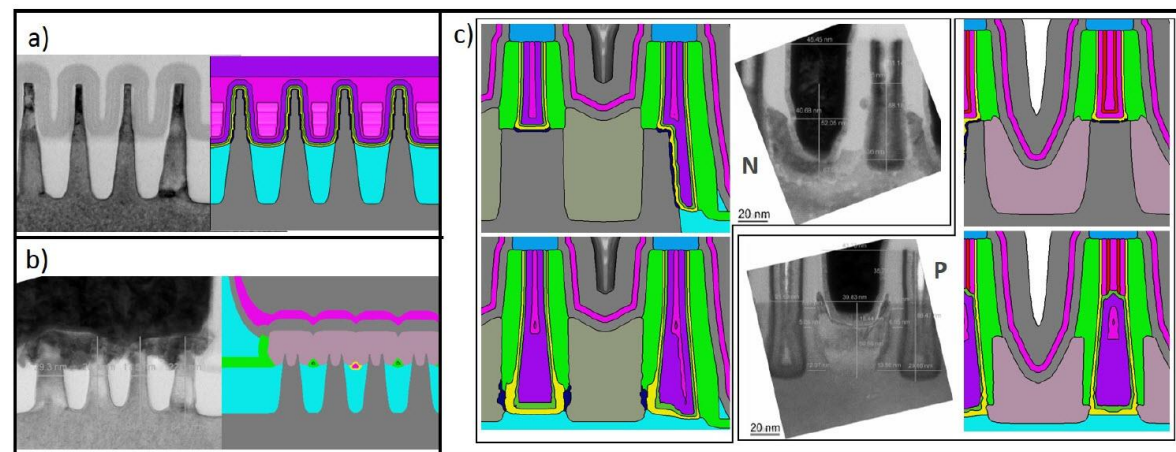
- ❑ Process variation causes delay variation in signal propagation.
- ❑ Inter-die variation leads to minor characteristic differentiations.
- ❑ Minor characteristic differentiations can be aggregated into major functional variations detectable by fuzzy logic.
- ❑ PUF is a derivative of such phenomenon.



<http://www.vlsi-expert.com/2011/02/process-variation-effects-on-design.html>



<https://www.semanticscholar.org/paper/From-Process-Corners-to-Statistical-Circuit-Design-Dongaonkar-Mudanai/3d734e03e8eea17851c8435a13433a619d35677b>



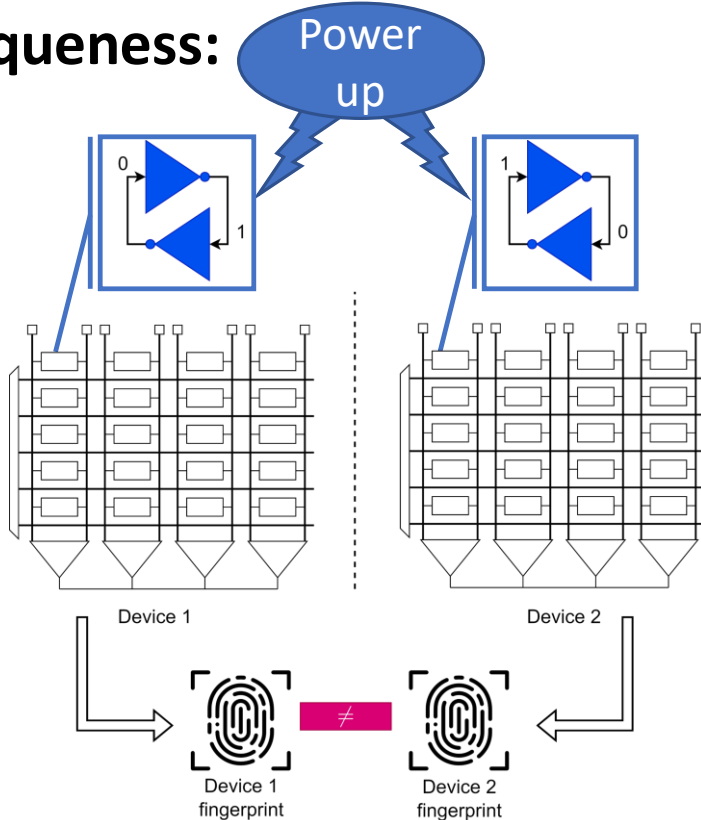
https://www.researchgate.net/figure/Global-variation-and-local-variation-For-local-variation-the-variance-in-length-depends_fig8_2982185

<https://www.coventor.com/paper/process-variation-analysis-of-device-performance-using-virtual-fabrication-methodology-demonstrated-on-a-cmos-14-nm-finfet-vehicle/>

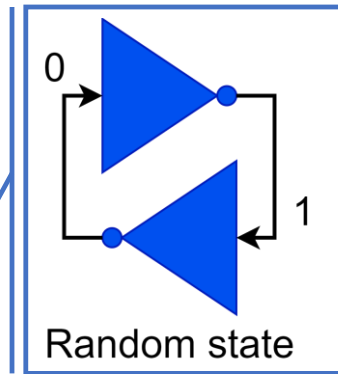
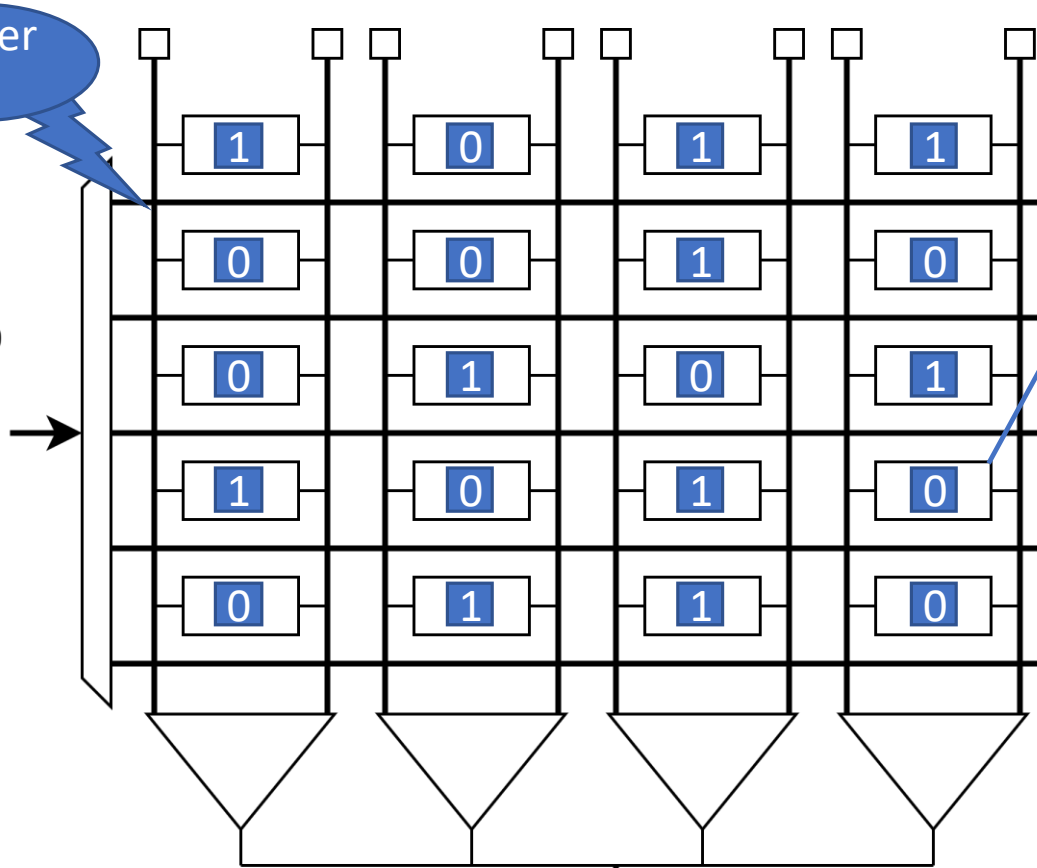
Ch2: Categories: Weak PUF: Example

- Weak PUF: Generates few CRPs (Often only 1 CRP with large dimensionality)
- Example: Memory based PUF
 - Example: Power up values of memory cells before firmware allocation
 - Limited number of fingerprints

Uniqueness:



Challenge



At device power up

Response

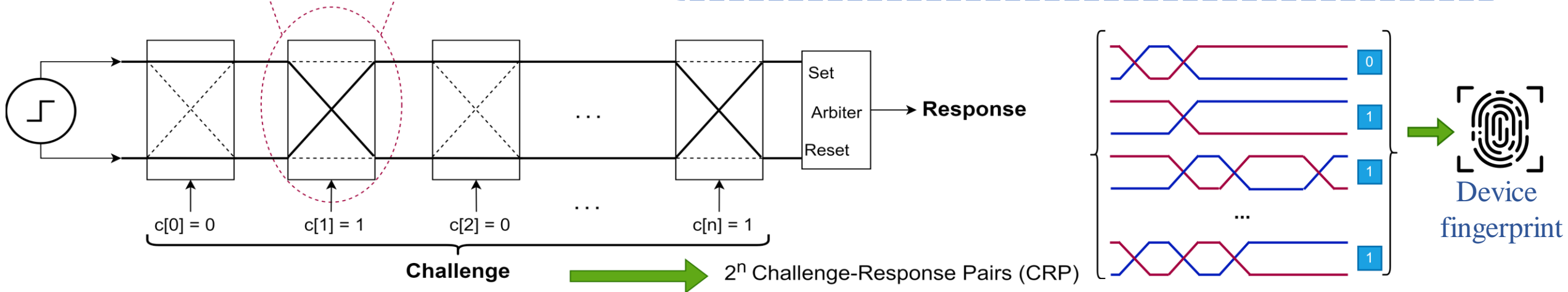
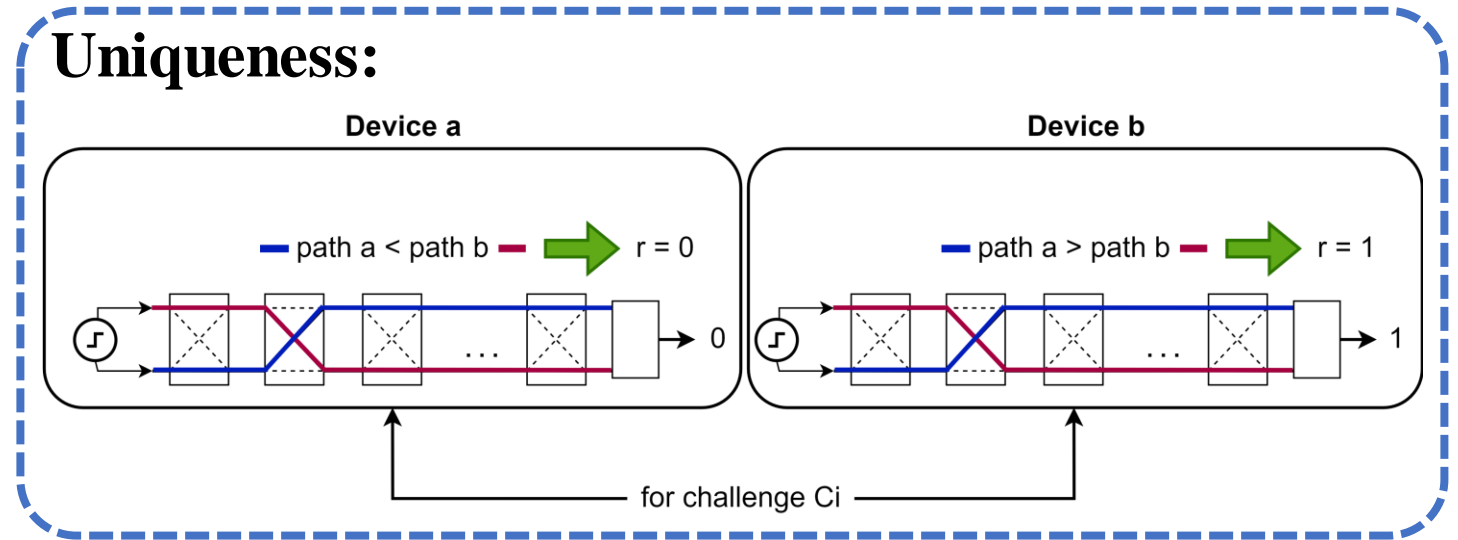
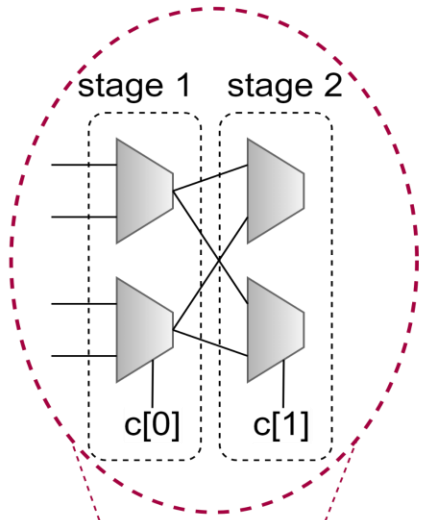
1	0	1	1
0	0	1	0
0	1	0	1
1	0	1	0
0	1	1	0

= [Fingerprint Icon] Device fingerprint

Ch2: Categories: Strong PUF & Example

➤ Strong PUF: Generates very large number of CRPs.

➤ Example: Arbiter PUF:



Ch1: Introduction: Commercial PUFs

❑ Intrinsic ID

- ❑ Butterfly PUF: Apollo FPGA IP (hardware IP)
- ❑ SRAM PUF: QuiddiKey IP (hardware IP)
- ❑ <https://www.intrinsic-id.com/physical-unclonable-function/>

❑ Enthentica

- ❑ FPGA PUF IP
- ❑ ASIC PUF IP
- ❑ <https://www.enthentica.com/about>

❑ PUFSecurity

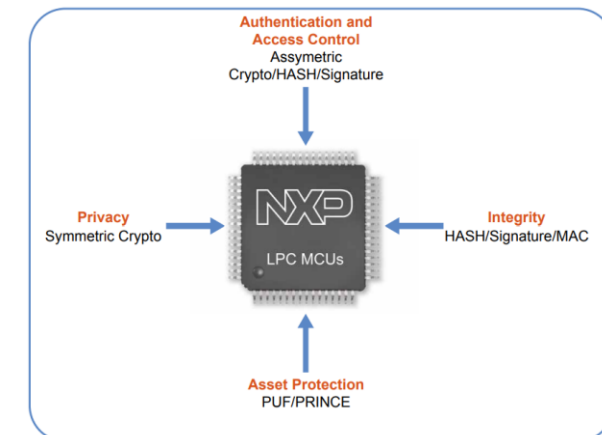
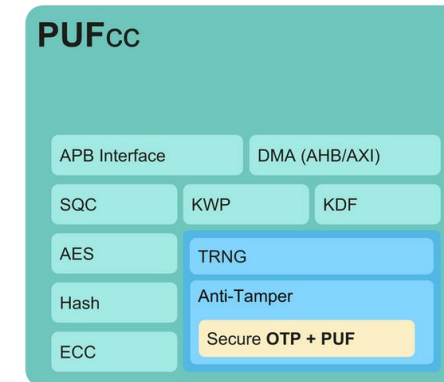
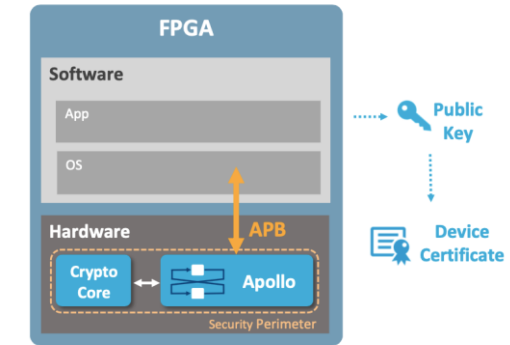
- ❑ PUF as root of trust (PUFrt)
- ❑ PUF as part of Crypto Processor (PUFcc)
- ❑ PUF as a Secure Element (PUFse)
- ❑ <https://www.pufsecurity.com/puf>

❑ NXP

- ❑ Multiple PUF-based solutions
- ❑ Check MCUXpresso SDK API
- ❑ https://mcuxpresso.nxp.com/api_doc/dev/2194/a00217.html

❑ Secure-IC

- ❑ Securyzr
- ❑ PUF as a root of trust element
- ❑ <https://www.secure-ic.com/products/issp/root-of-trust/>



Ch2: Features of a Good PUF

Randomness [1]

- ❑ Good randomness: when 0s and 1s in a CRP set are equally distributed.
- ❑ Can also be described as uniformity of the 1s and 0s in a CRP set.

$$H = -\log_2 \max(p, 1 - p)$$

Frequency of 1s

$$p = \frac{1}{N_r} \sum_{i=1}^{N_r} b_i$$

Total number of response bits

*i*th response bit in the set of all response bits

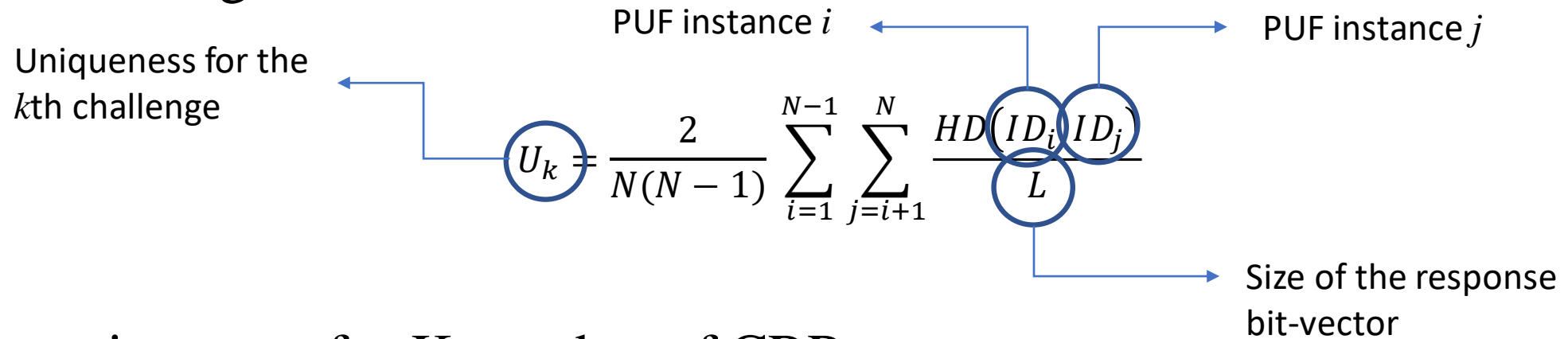
❑ Ideal value for p is 0.5

❑ Consequently, the value of H is between 0 to 1. Ideal is close to 1.0

Ch2: Features of a Good PUF

Uniqueness [2]

Measures the uniqueness of PUF responses in-between instances of the same PUF circuit design.



Average uniqueness for K number of CRPs:

$$MU = \frac{1}{K} \sum_{i=1}^K U_k$$

Average uniqueness is normalized between 0 to 1. Ideal value is 1.

Ch2: Features of a Good PUF

Diffuseness[3]

Measures how different are the set of responses from the same PUF instance to different challenges.

Size of the response bit-vector

Number of CRPs

Response bit

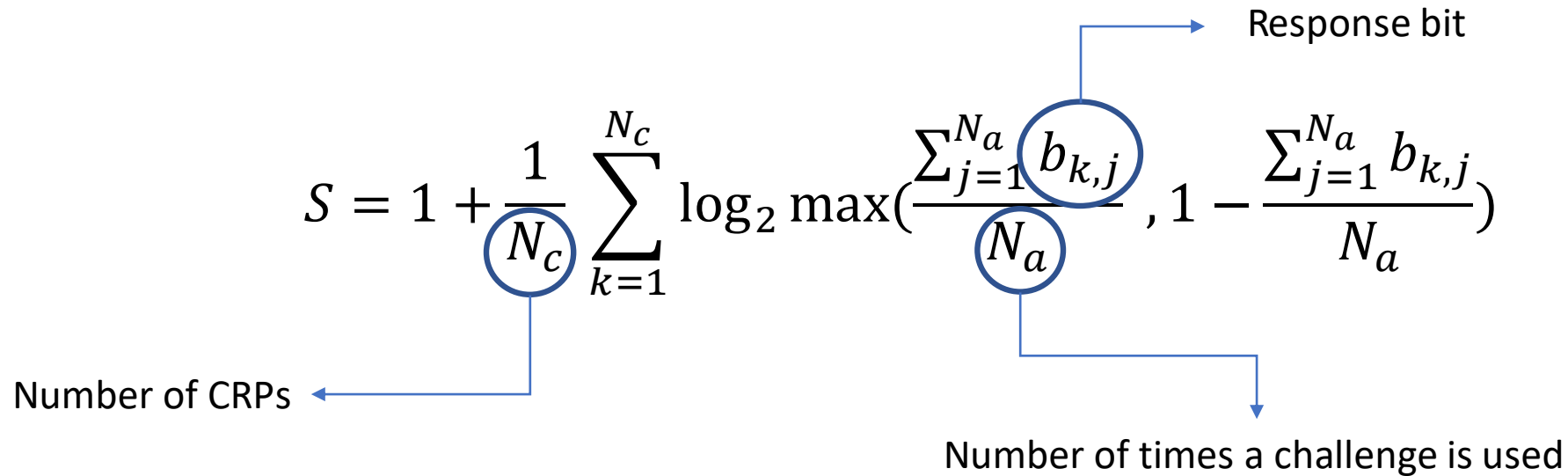
$$D = \frac{4}{K^2 \times L} \sum_{l=1}^L \sum_{i=1}^{K-1} \sum_{j=i+1}^K (b_{i,l} \text{ XOR } b_{j,l})$$

Diffuseness is normalized between 0 to 1. Ideal value is 1.

Ch2: Features of a Good PUF

Reliability[4]

- ❑ Measures the stability of the response given for each challenge in multiple acquisitions.
- ❑ Its measured for a CRP set.

$$S = 1 + \frac{1}{N_c} \sum_{k=1}^{N_c} \log_2 \max\left(\frac{\sum_{j=1}^{N_a} b_{k,j}}{N_a}, 1 - \frac{\sum_{j=1}^{N_a} b_{k,j}}{N_a}\right)$$


Number of CRPs

Response bit

Number of times a challenge is used

- ❑ The steadiness is measured between 0 to 1 and the ideal value is 1.

Ch2: Verification of Design using Simulation

Benefits of using PUF simulation:

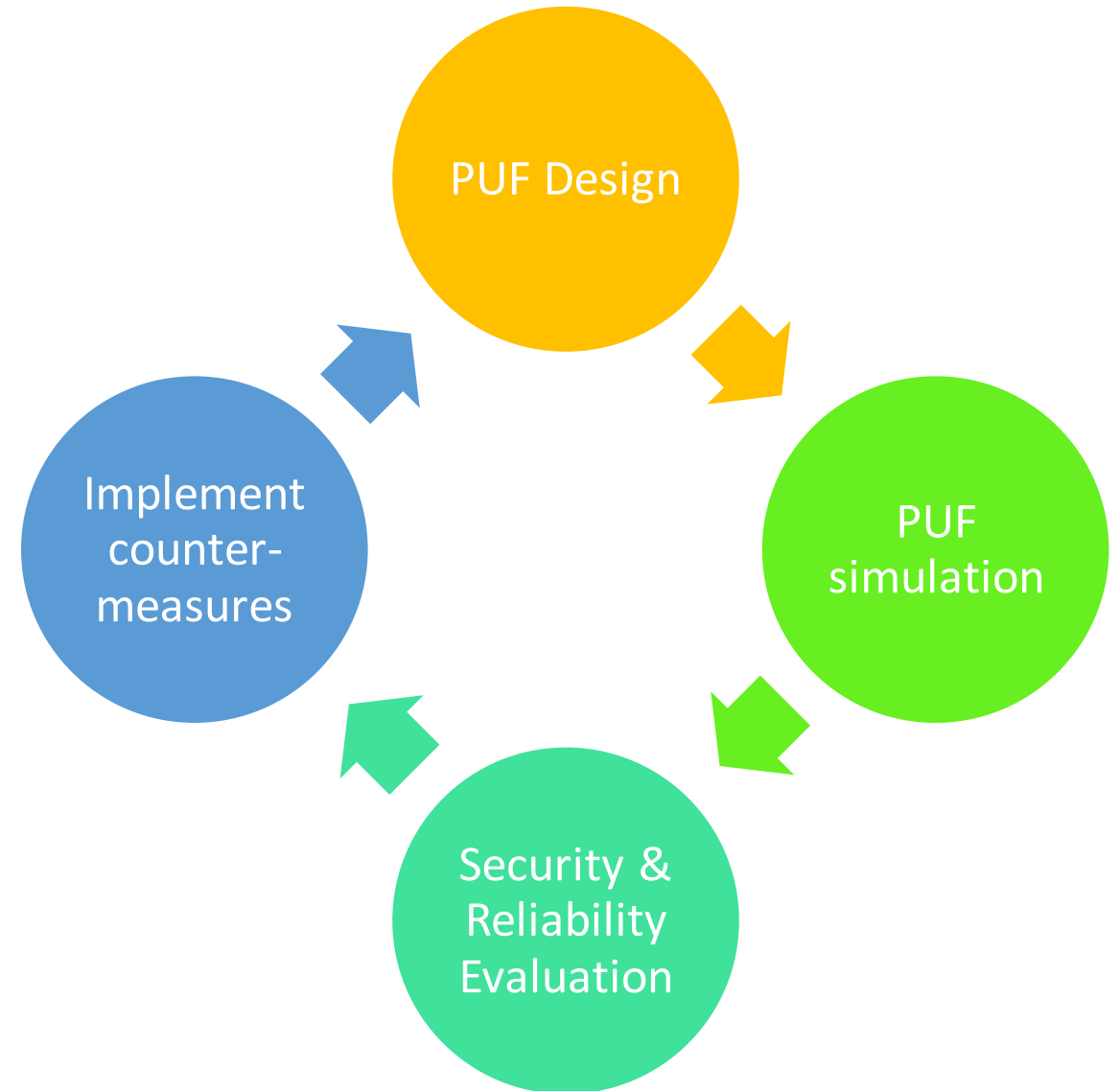
Design Flexibility:

- Allows Rapid testing
- Allows early detection of security or reliability flaws

Existing simulations

(See [Appendix 1](#))

- Ruhrmair's LR PUF simulation source[5]
 - Arbiter and XOR Arbiter and Lightweight Arbiter PUF simulator
 - Link: <http://www.pcp.in.tum.de/code/lr.zip>
- PyPUF simulator source[6]
 - Simulator for variations of Arbiter PUF
 - Simulator for Bistable Ring PUF and compositions
 - Link: <https://pypuf.readthedocs.io/en/latest/>



Ch2: Assignment

Using Python or Matlab: Simulate a 64 stage Arbiter PUF

1- Use random number generator to generate the delay values of each element in the design:

- Multiplexer
- Parallel Paths
- Crossing Paths

Make sure the delay values for each element class are normally distributed

Recommendation: For the random number generator use mean 0 and variance 1.

2- Connect the elements accordingly to build the main body of the PUF.

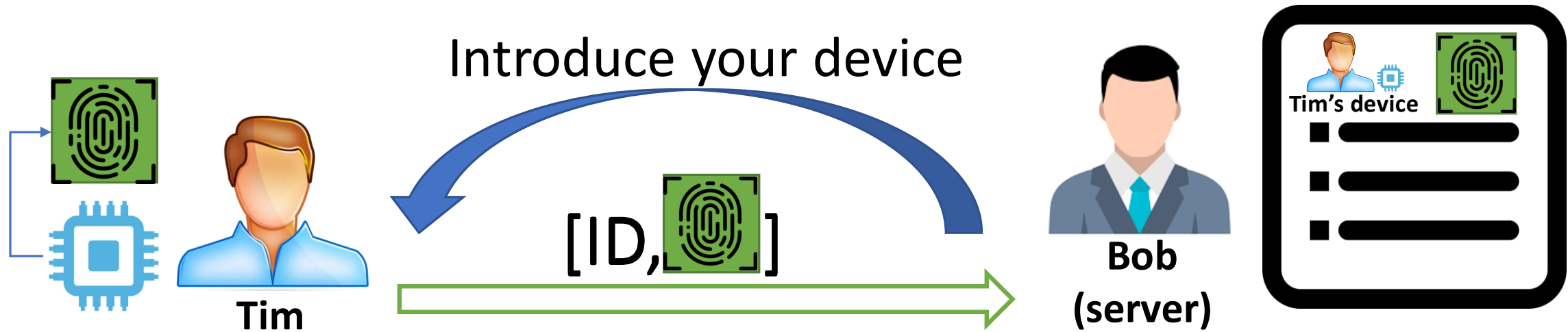
3- Develop a comparator at the end which compares the accumulated delay values of the two competing paths. This is the arbiter at the end of the PUF.

4- Generate 10 instances using the simulator code.

5- Then generate 10,000 CRPs for each instance and measure the Randomness and Diffuseness of each instance, and measure the Uniqueness of the PUF group.

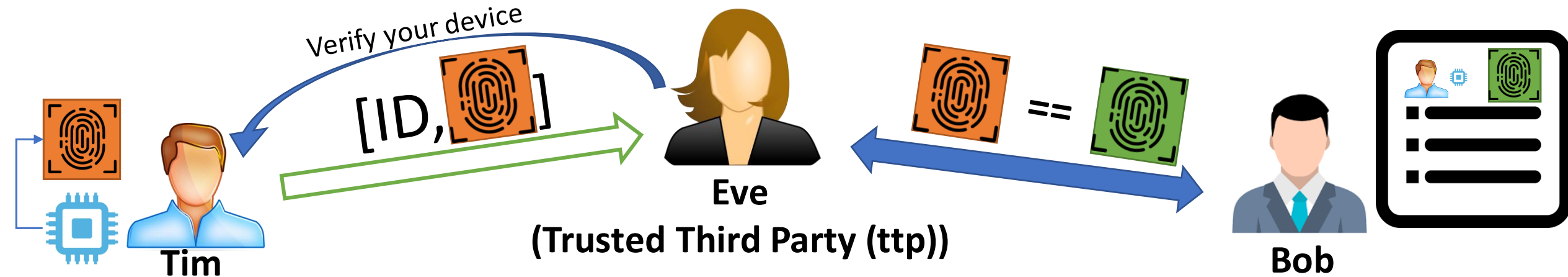
Ch3: PUF Protocols: Authentication

- Enrollment: Saving CRPs of a PUF on server
- Authentication: Exchanging CRP to find a match on CRP database



Ch3: PUF Protocols: Authentication

- Enrollment: Saving CRPs of a PUF on server
- Authentication: Exchanging CRP to find a match on CRP database



Ch3: PUF Protocols: Authentication: Example

- Mutual authentication with exchanging only challenge values [7]

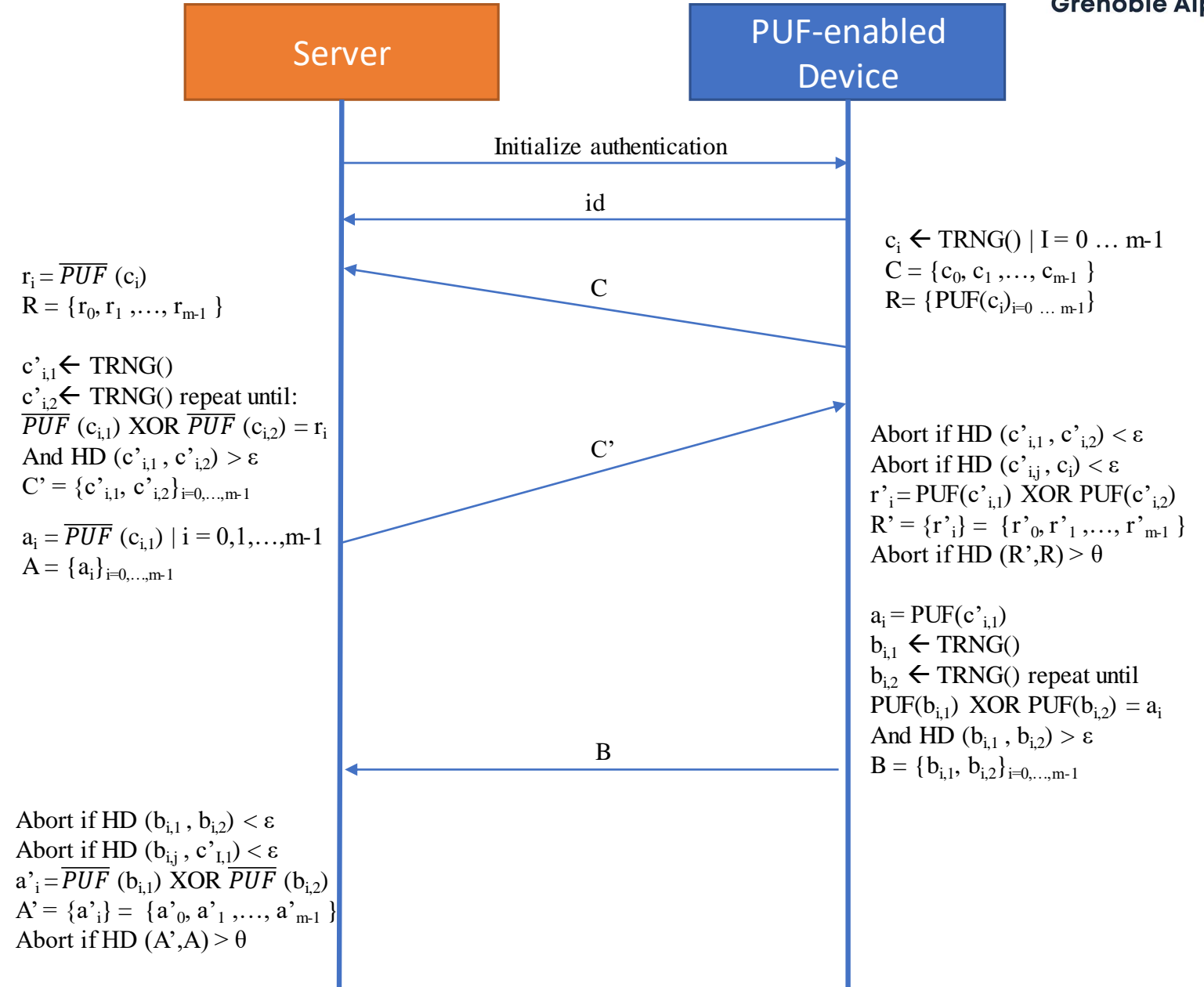
➤ Mutual authentication

- PUF-enabled device authenticates the server
- Then, the server authenticates the PUF-enabled device

➤ \overline{PUF} is an equivalent model of PUF which exists on server.

- This model provides access to all the CRPs that PUF can generate

- $\underline{\epsilon}$ and $\underline{\theta}$ are minimum acceptable thresholds of hamming distance between challenge values \underline{C} for server authentication and challenge values \underline{B} for PUF-enabled device authentication, respectively.



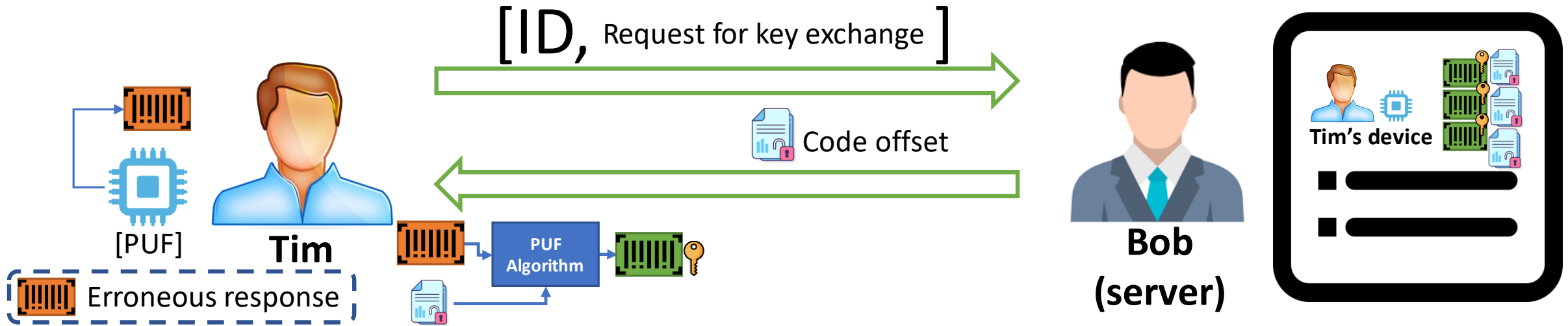
Ch3: PUF Protocols: Key Generation

- Enrollment: Save the initial readout of PUF as encryption key
- Key exchange: Deliver code offset of the encryption key to Tim to recover the original key

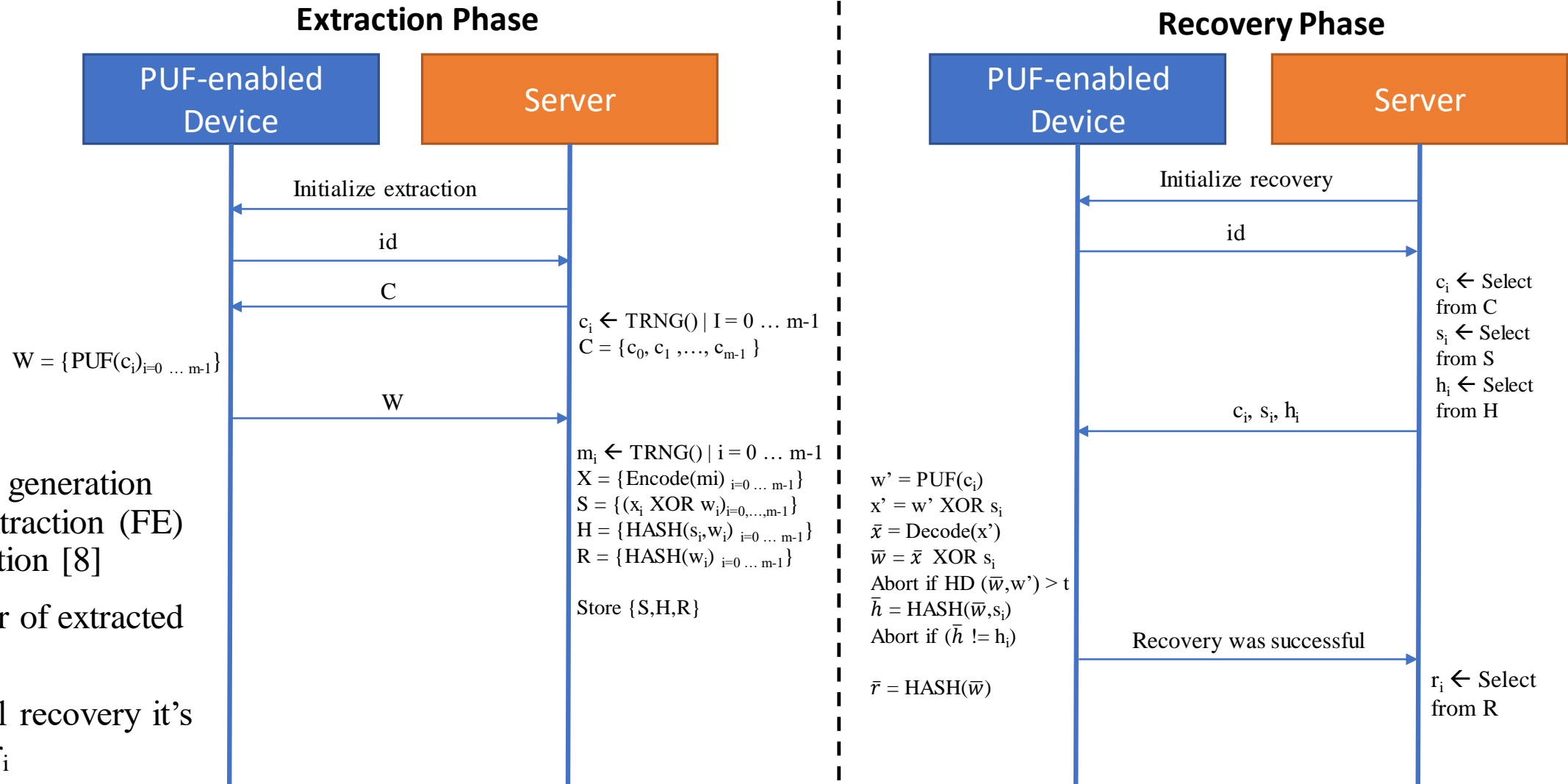


Ch3: PUF Protocols: Key Generation

- Enrollment: Save the initial readout of PUF as encryption key
- Key exchange: Deliver code offset of the encryption key to Tim to recover the original key



Ch3: PUF Protocols: Key generation: Example



- PUF-based key generation using Fuzzy Extraction (FE) for error correction [8]
- m is the number of extracted keys from PUF
- After successful recovery it's expected $\bar{r} == r_i$
- \bar{r}_i and r_i are encryption keys

Ch3: Assignment

Using the previously developed PUF simulator code, now Develop an authentication protocol.

Before that, create a noise simulator which adds noise to the PUF CRP.

NOTE: A noise in PUF CRP ultimately leads to flipping the true response value for a given challenge.

Then develop the authenticator which communicates with each PUF instance and reads their CRP which is coped with noise. The Authenticator should be able to:

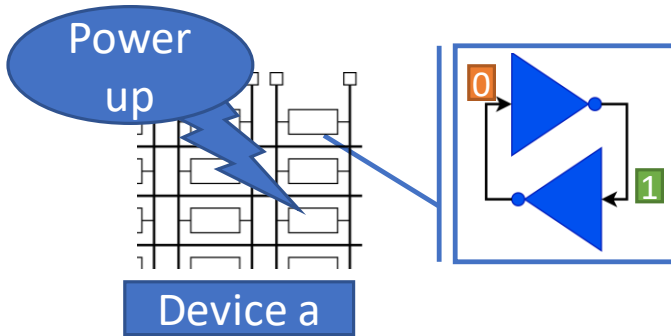
- 1- Enroll first a certain amount of CRPs.
- 2- In mission mode, request for CRP from the PUF and compare the oracle CRP with the re-captured CRP and vote for the authenticity.

Measure the success-rate of authentication for each device.

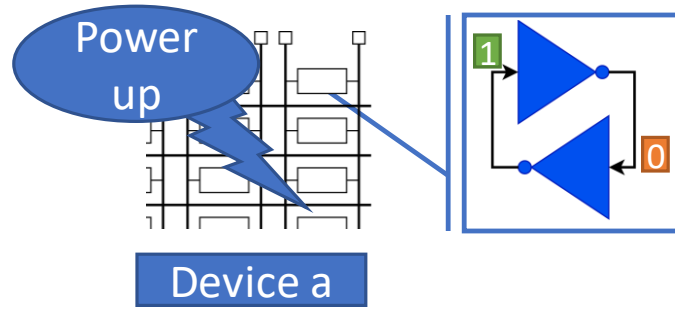
Ch4: Challenges: Instability

Fluctuation of response value over multiple close acquisitions

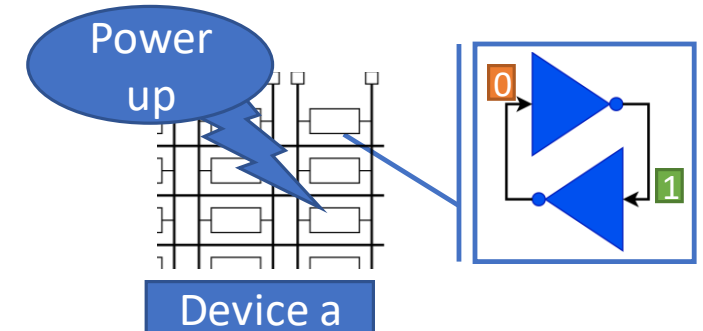
At time t :



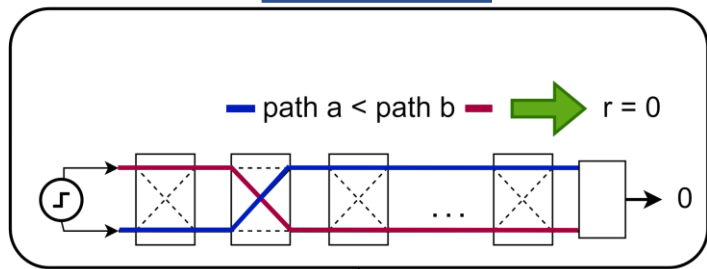
At time $t + \Delta t$:



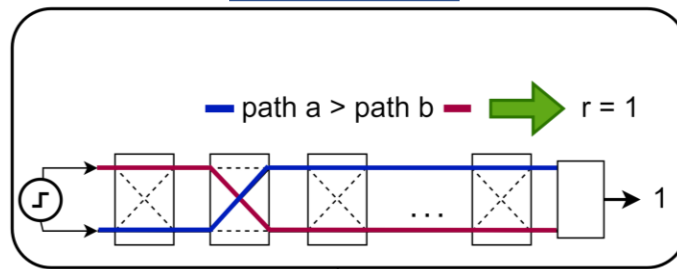
At time $t + \Delta t + \Delta t$:



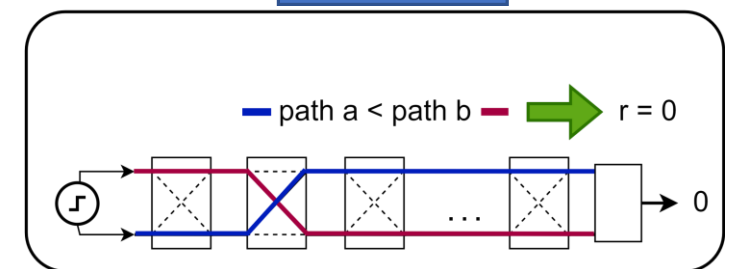
Device a



Device a



Device a

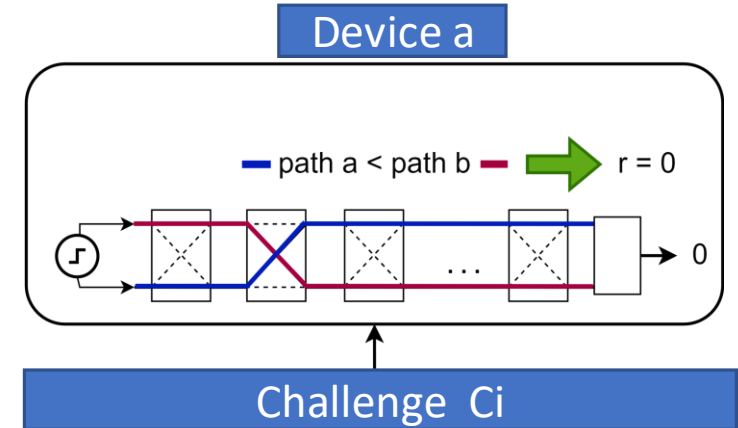
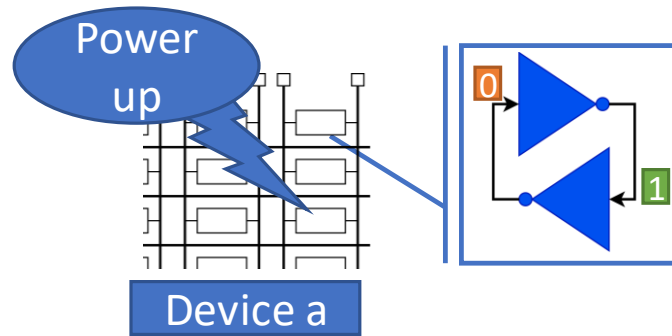


Ch4: Challenges: Aging

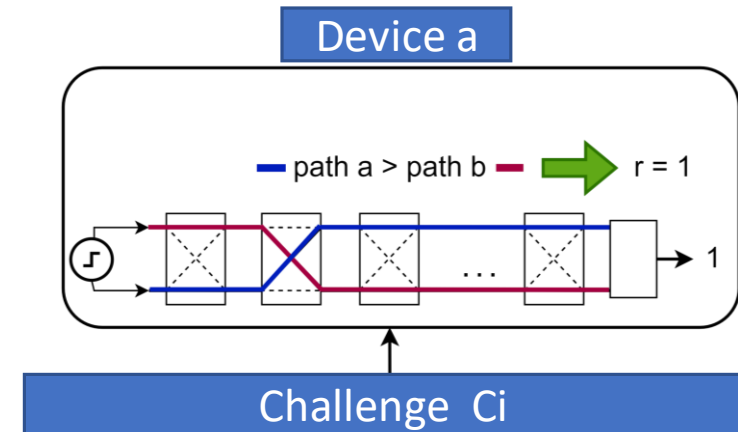
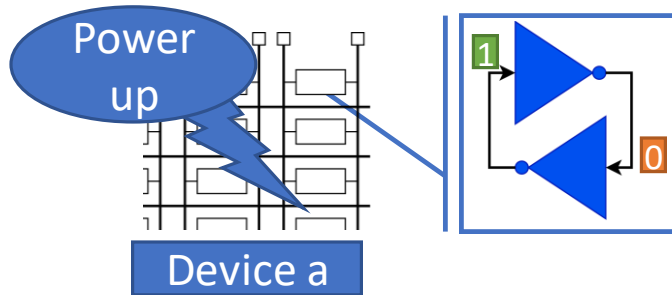
Transition from one state to other over a period of time.

At $t + \Delta t$ and over, the state of the response is changed for a long term.

At time t :

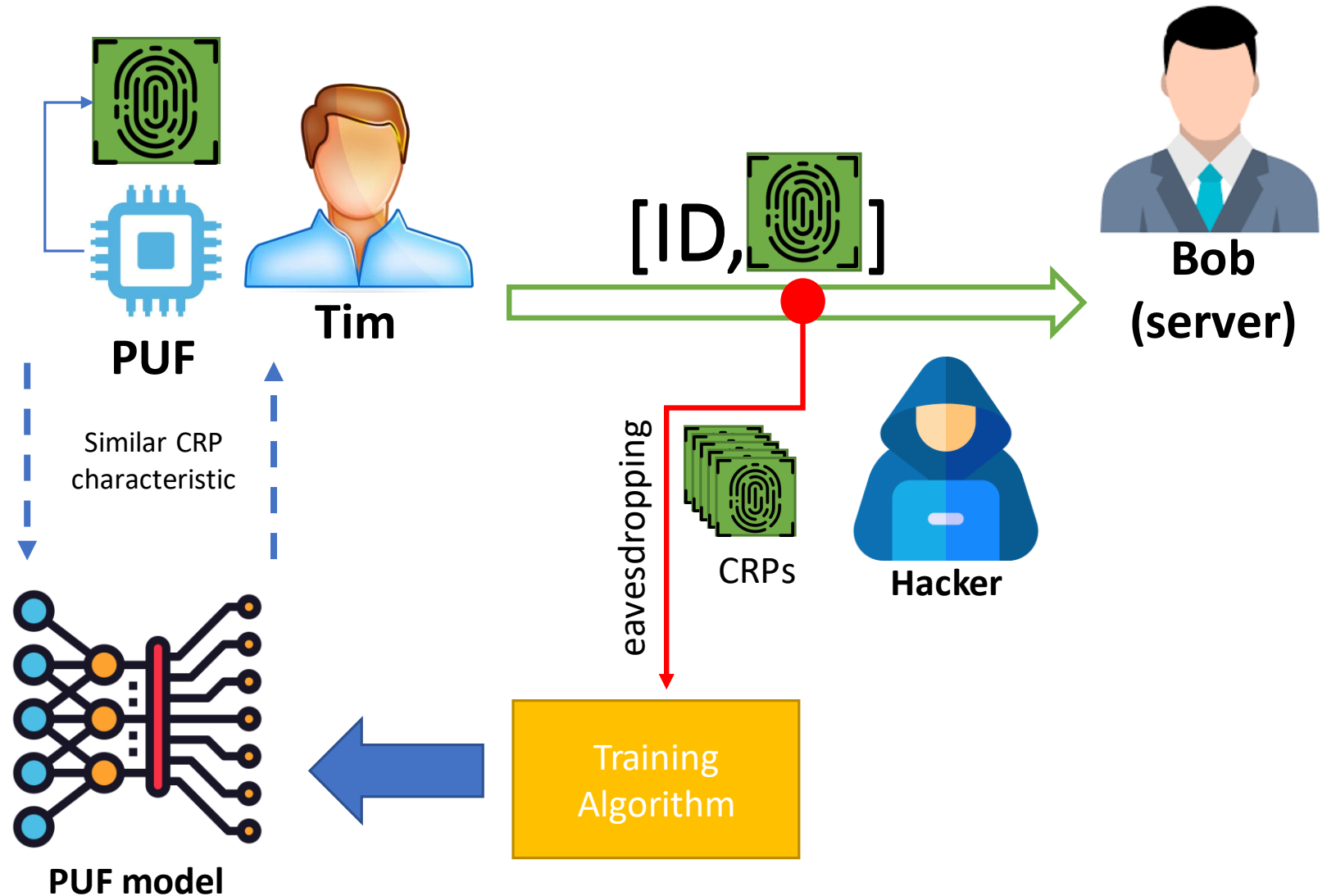


At time $t + \Delta t$:



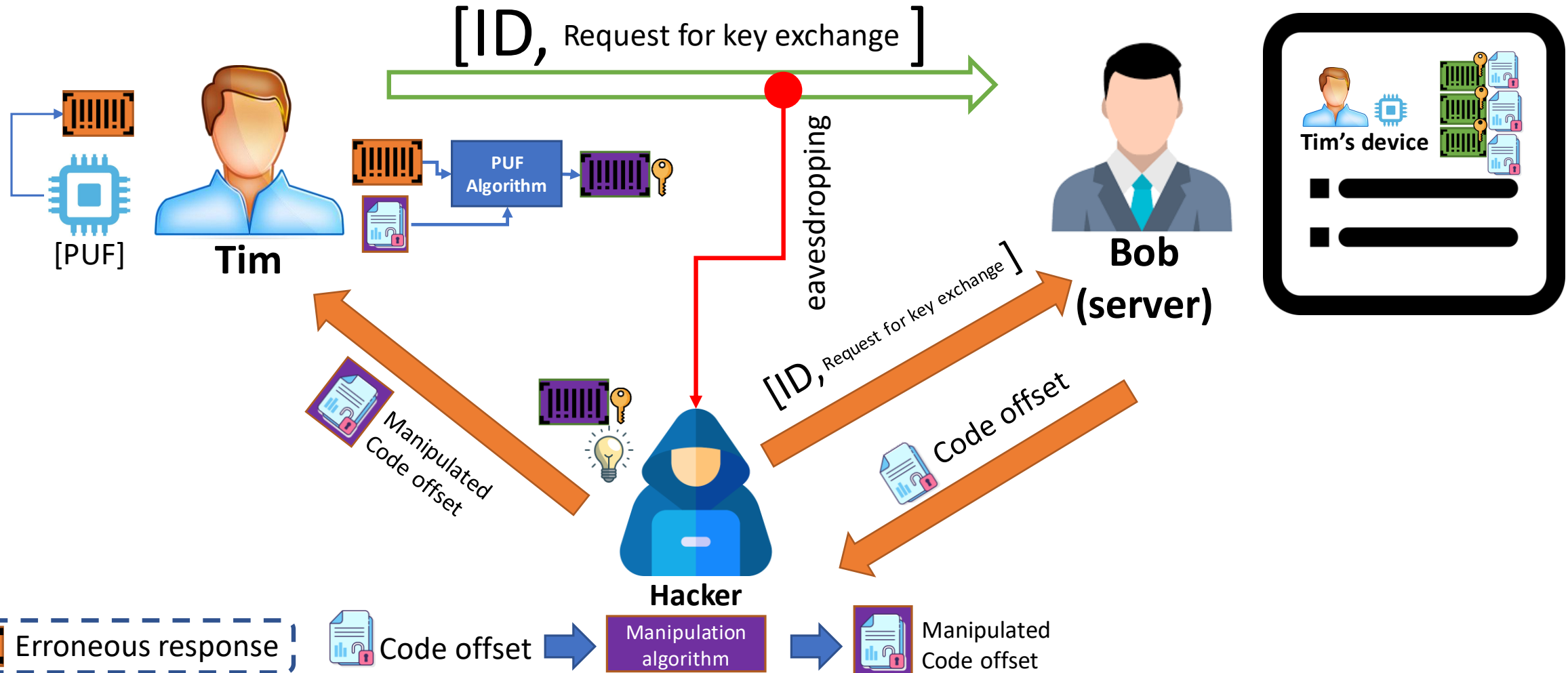
Ch4: Challenges: Security: Cloning Attack

- ❑ Goal is to build a digital clone of the PUF[9]
- ❑ Using the PUF model, attacker can impersonate the authentic user (Tim)
- ❑ PUF model will have some error in mimicking the PUF
- ❑ Goal of training algorithm is to reduce the mimicking error
- ❑ Depending on the accuracy goal and PUF complexity, number of required CRPs for training can differ



Ch4: Challenges: Security: Helper Data Manipulation Attack

- The goal of the attack is to make the guessing work of the encryption key easier by manipulating the code offset and redirecting it to the PUF owner [10]



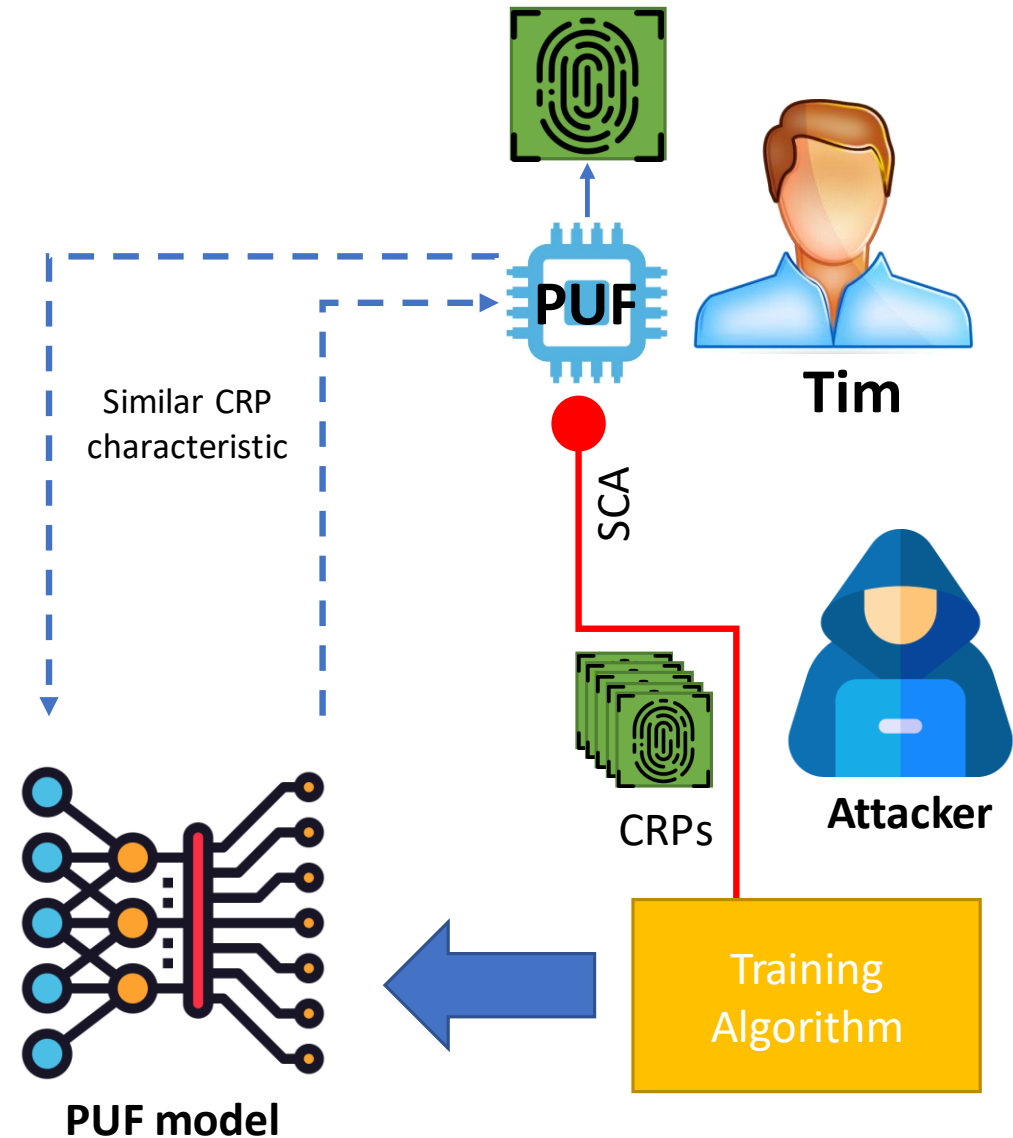
Ch4: Challenges: Security: Side Channel Attack

- Similar to cloning attack, the goal is to build a digital clone of the PUF[11]
- Using the PUF model, attacker can impersonate the authentic user (Tim)
- The medium of auditing to capture CRP is through side channel analysis
- Various Side channel analysis options
 - Power tracing
 - Electromagnetic emission tracing
 - Etc.



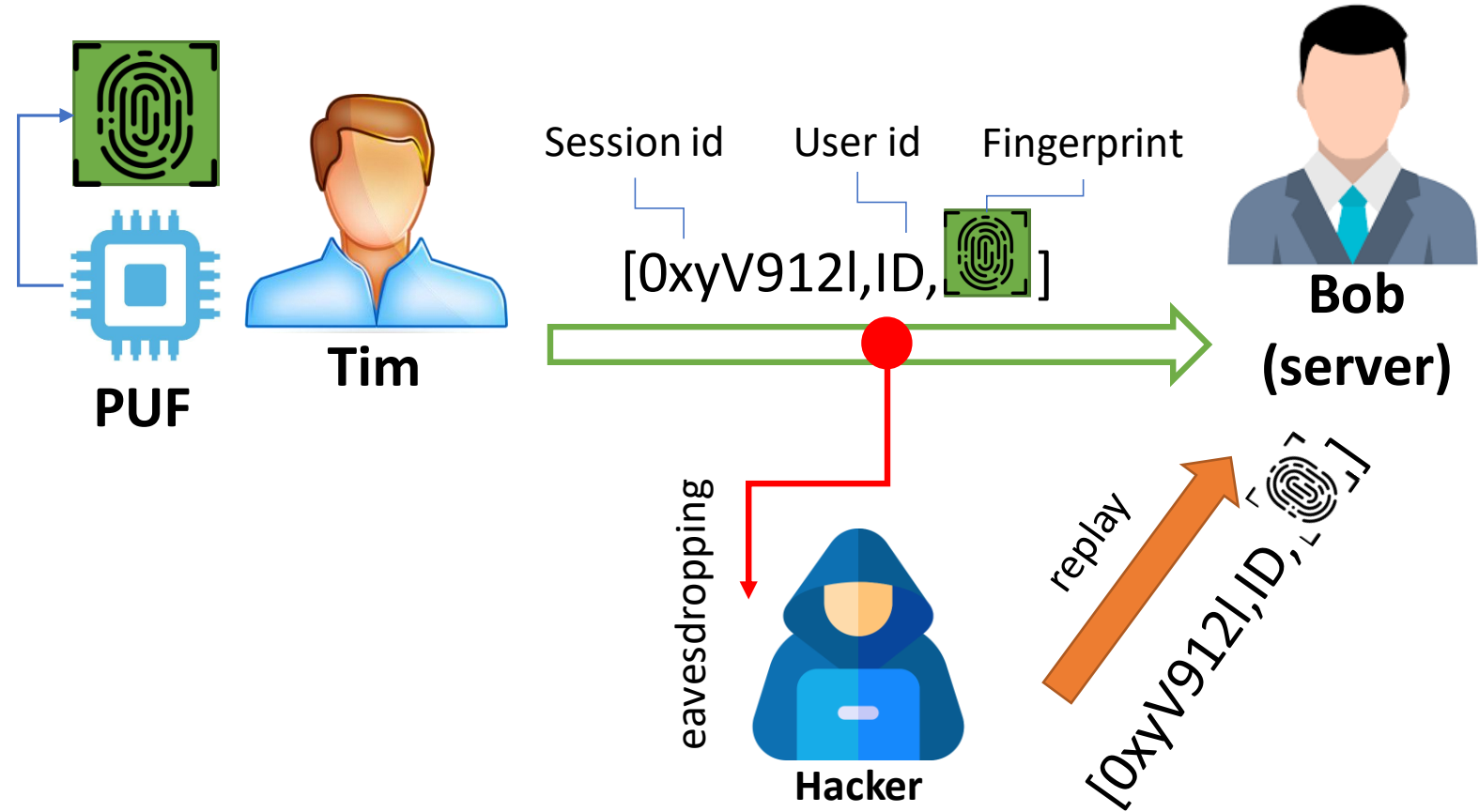
<https://www.esynov.fr/accueil/banc-de-test-1>

Example of side channel analysis equipment



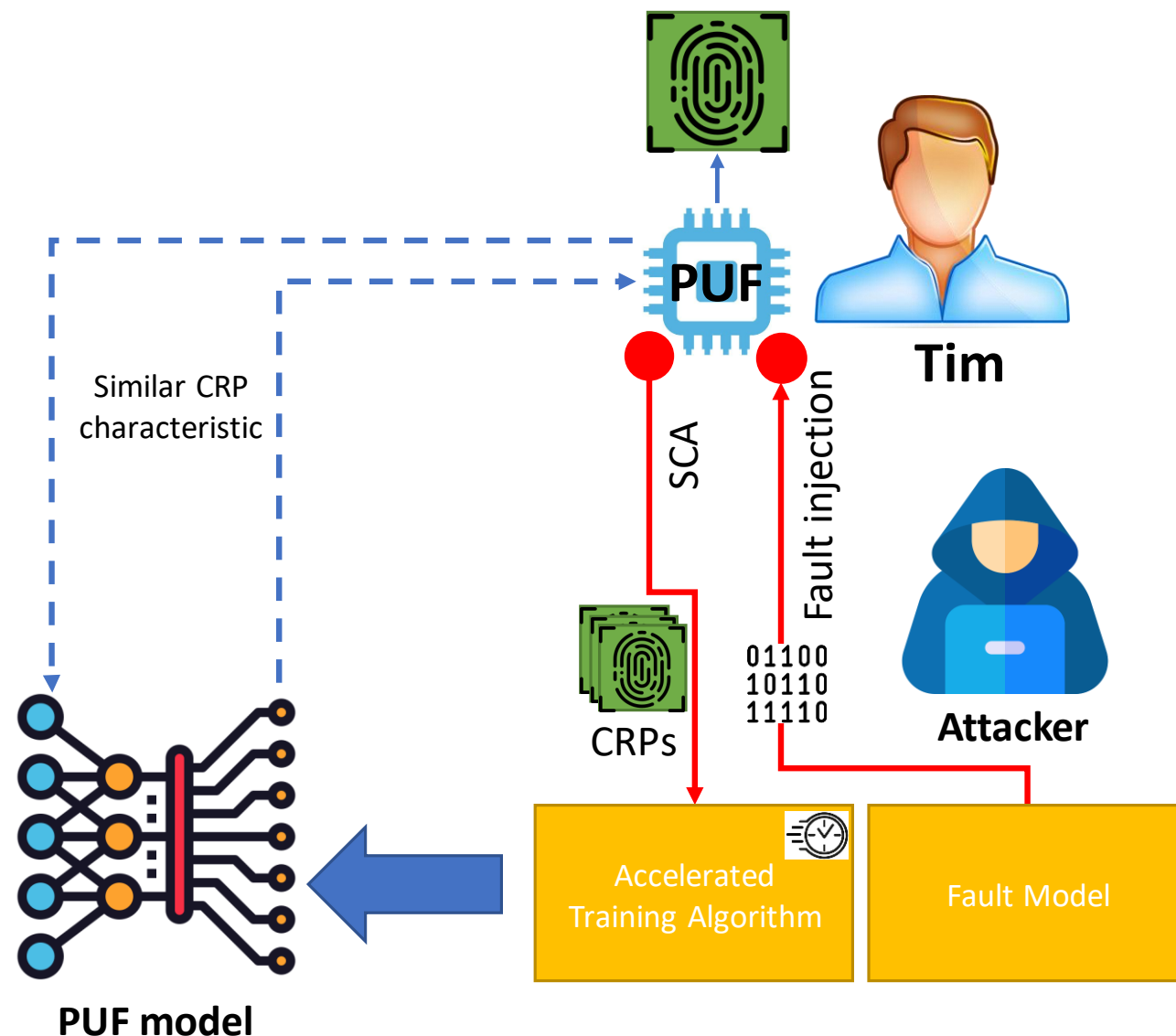
Ch4: Challenges : Security: Replay Attack

- The goal of the attack is to reuse the transmitted authentication packet to impersonate the PUF owner (Tim). [12]
- To prevent such attack, each time a new session is created and a new fingerprint is used.
- To prevent replay attack, a large dataset of fingerprints needs to be provided for the server.



Ch4: Challenges: Security: Fault Injection Attack

- Similar to cloning attack and side channel attack, the goal is to build a digital clone of the PUF[13]
- Using the PUF model, attacker can impersonate the authentic user (Tim)
- The main goal in this attack is to use fault injection to accelerate the training process.
- Assumed that after fault injection, fewer CRPs will be required to generate the PUF model.



Ch4: Assignment (optional)

Using the Arbiter PUF simulator developed in previous assignments:

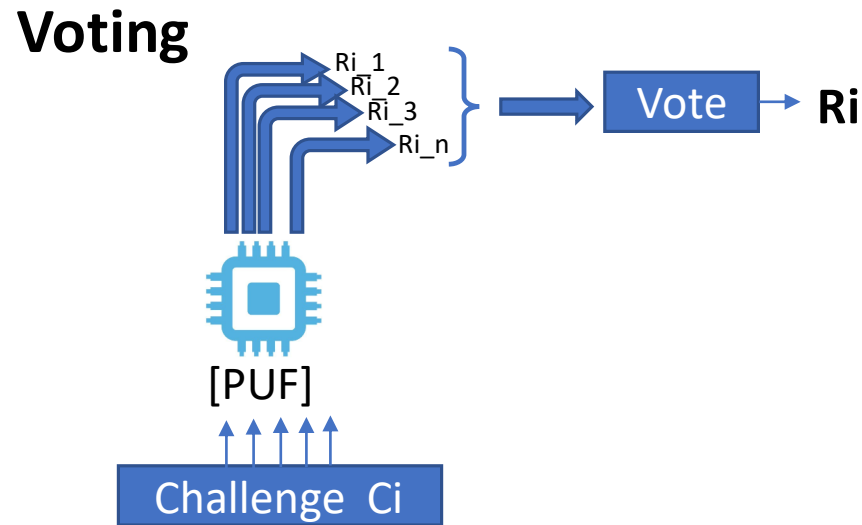
Capture new CRP dataset from a selected PUF instance and attempt in modeling the PUF using Python Machine Learning Libraries like Pytorch or Python (See Appendix 3 for suggested model specification).

Deliverables:

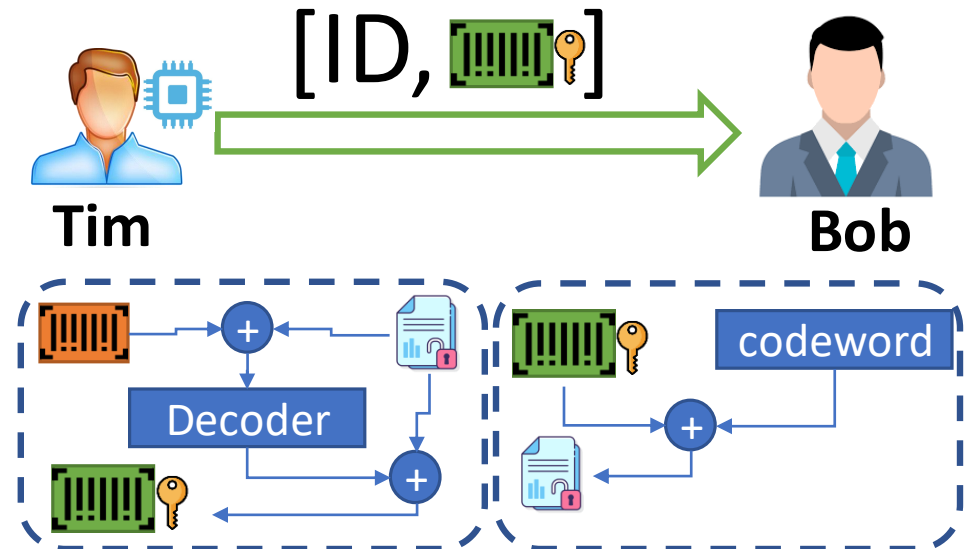
- 1- Create a predictive model and initialize its learning parameters and perform the training in adequate amount of steps until convergence.
- 2- Monitor the prediction accuracy, and learning loss value at each training epoch (training step). These values then need to be plotted (See Appendix 3 for an example plot).
- 3- Observe the final prediction accuracy, and time of training.
- 4- Repeat this for every PUF instance that was simulated previously.
- 5- Report the minimum number of CRPs you used to achieve prediction accuracy above 90%.

Ch5: Instability and Aging Solutions

- Soft error correction can be done by Voting. This is suitable for authentication
- Encryption keys need to be robust and error free. This requires robust error correction algorithms such as Fuzzy Extraction (FE).



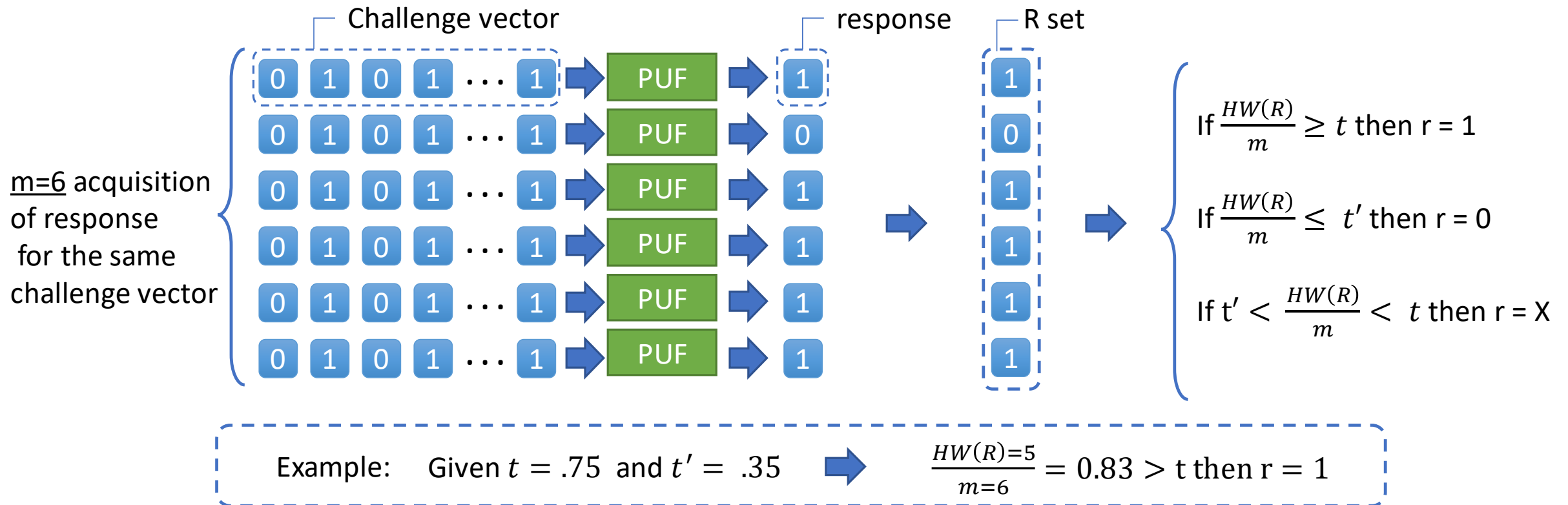
ECC + FE



Ch5: Instability and Aging Solutions

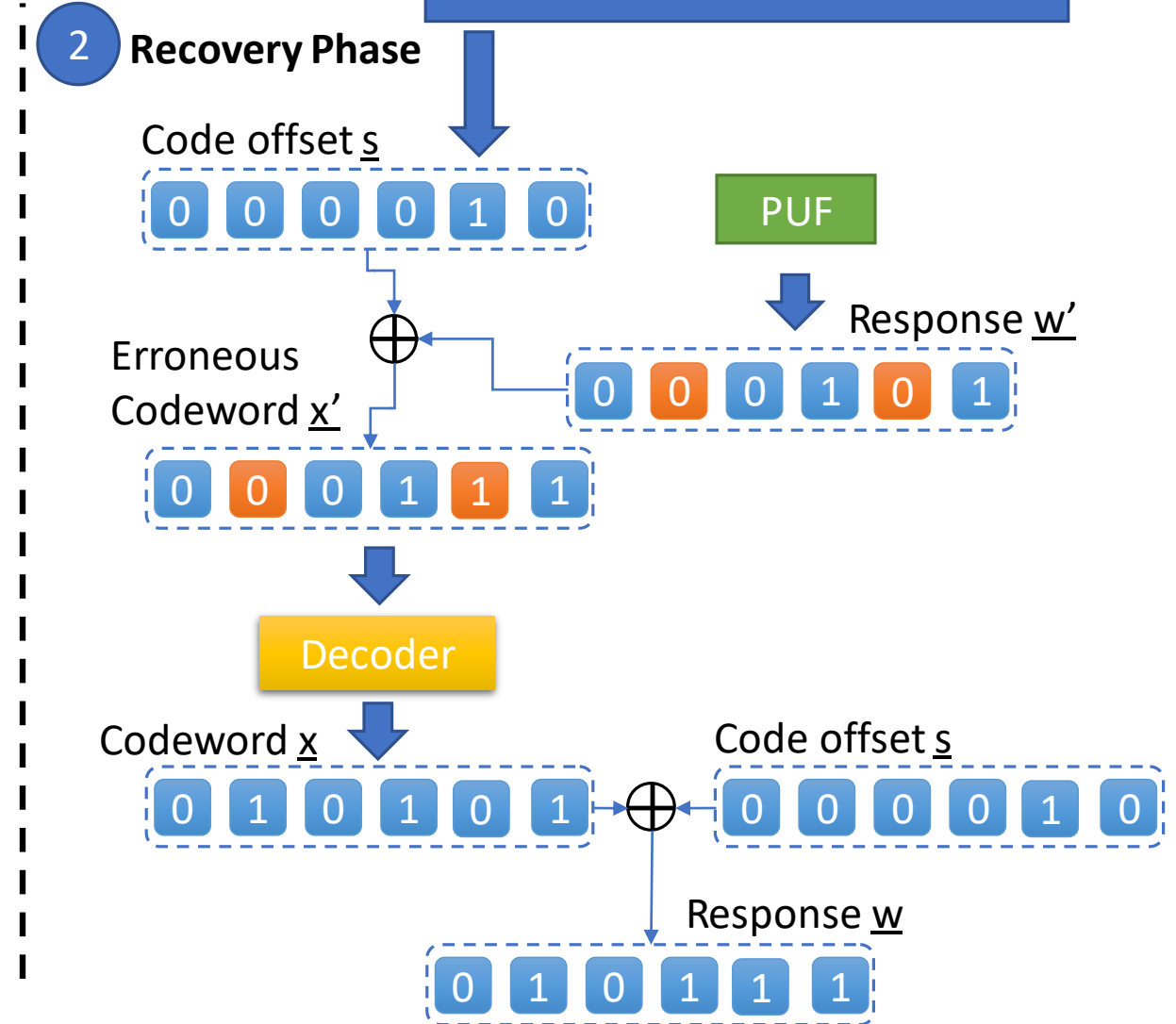
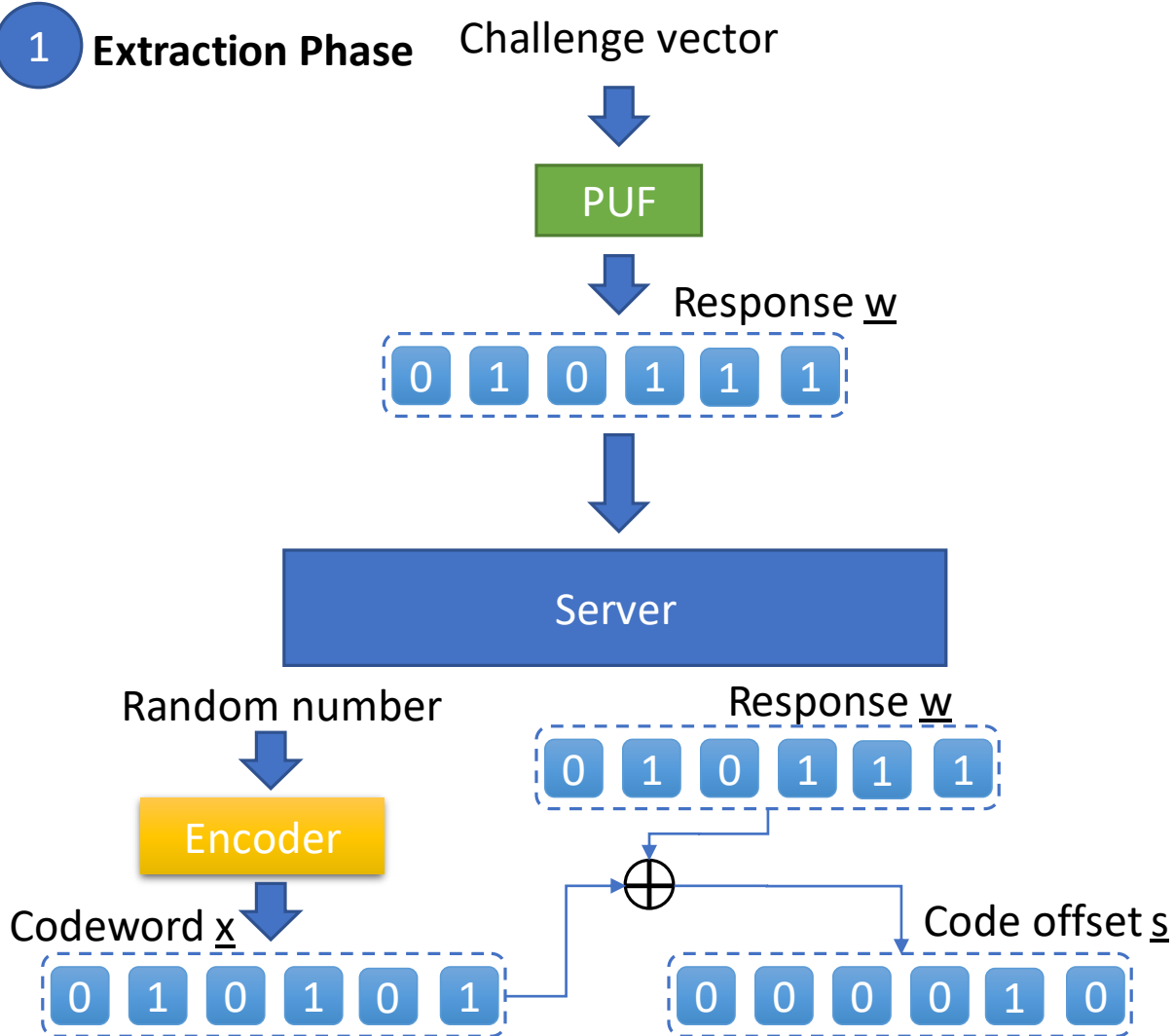
Example for Voting solution

- \underline{r} represents the voted value for the response of the given challenge
- \underline{t} is the threshold of Hamming Weight (HW) of \underline{R} set more than which means that \underline{R} represents response value $\underline{1}$
- \underline{t}' is the threshold of HW of \underline{R} set less than which means that \underline{R} represents response value $\underline{0}$
- $r = X$ means that the response is considered unstable for the given challenge. During enrollment, such CRPs are discarded.



Ch5: Instability and Aging Solutions

Example for ECC + FE solution



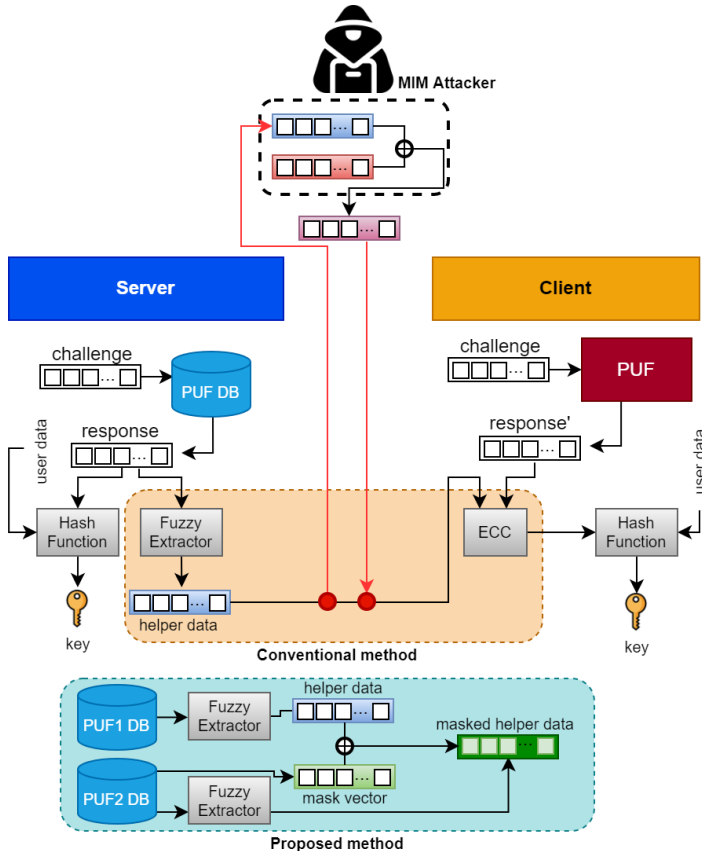
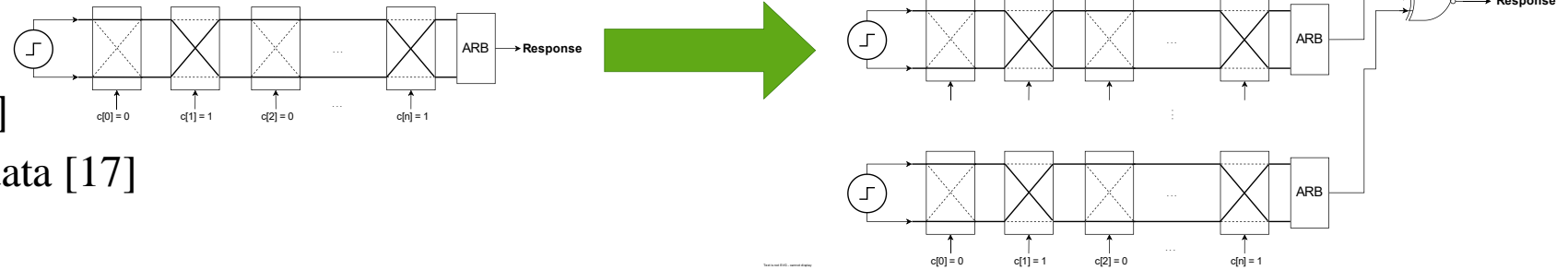
Ch5: Security Solutions and Countermeasures

❖ Strong PUF

- Increasing complexity [14]

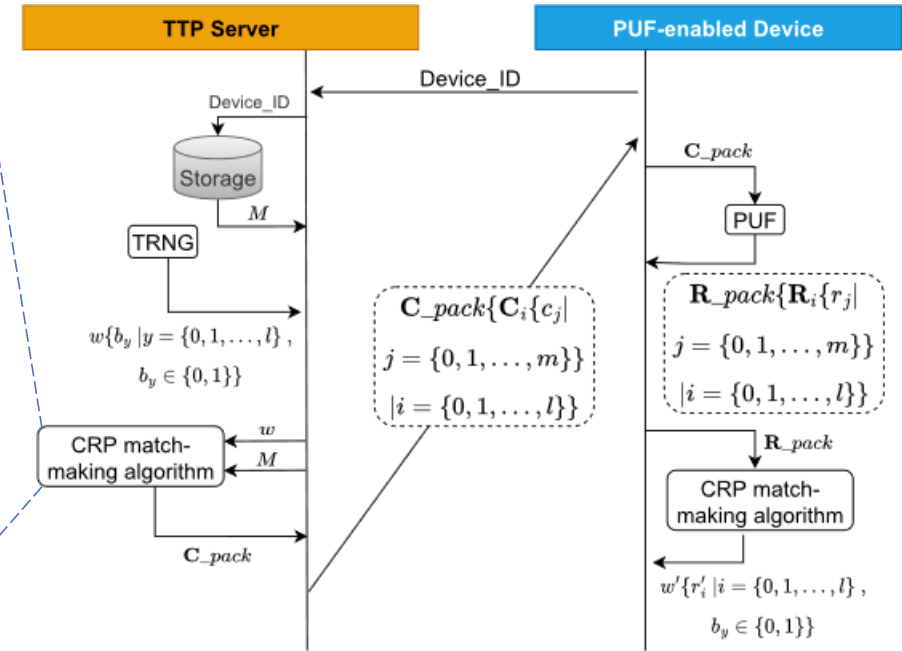
❖ Weak PUF

- Protecting Public data [15][16]
- Key recovery without helper data [17]



```

Require:  $w\{b_y | y = \{0, 1, \dots, l\}, b_y \in \{0, 1\}\}$ 
Require:  $M\{\Theta, \epsilon > t\}$ 
 $L \leftarrow l$ 
 $M \leftarrow m$ 
while  $i < L$  do
  while  $j < M$  do
     $ch \leftarrow$  randomly generate challenge vector
     $rp \leftarrow M(ch)$ 
    if  $rp = w[i]$  then
      if  $ch \notin C\_pack$  then
         $j \leftarrow j + 1$ 
         $C\_pack[i] \leftarrow ch$   $\triangleright$  Append  $ch$  to  $C\_pack$ 
      else
         $j \leftarrow j + 1$ 
    end while
   $i \leftarrow i + 1$ 
return  $C\_pack$ 
  
```



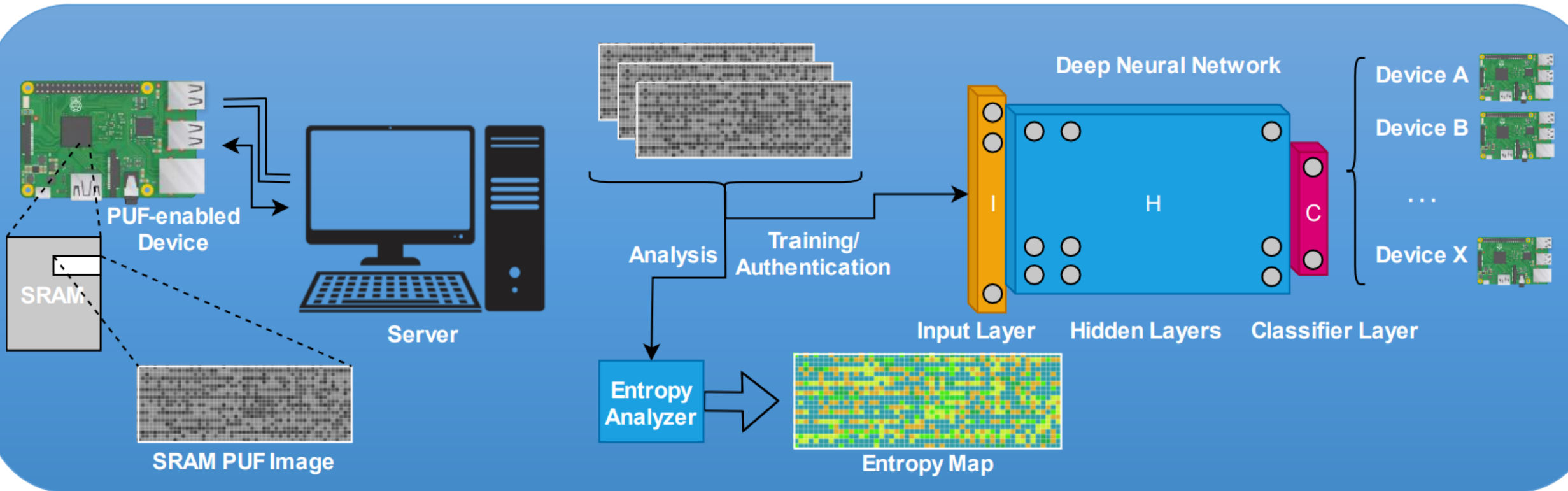
Ch6: Emerging Topics: ML and PUF Protocol

❑ Authentication: DLA PUF[18]

❑ PUF memory values are captured as PUF image

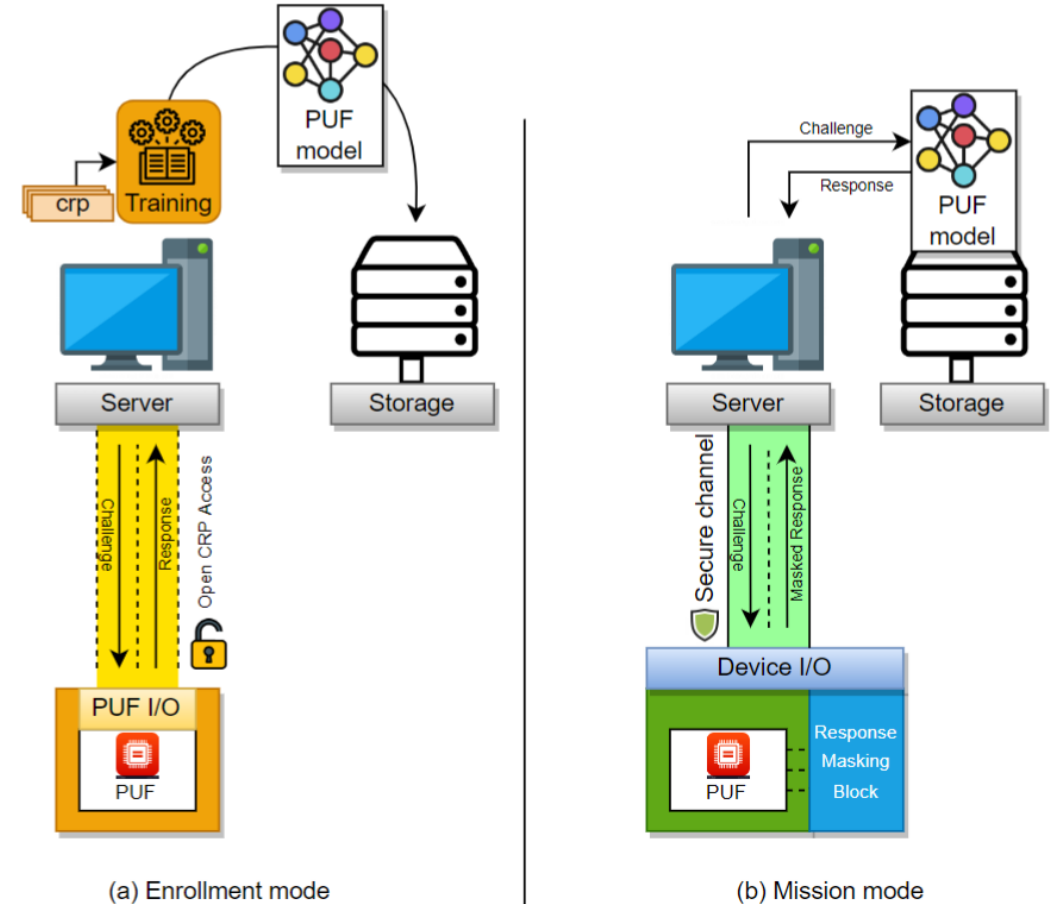
❑ Each device has its own PUF image

❑ The DNN model is responsible for recognizing the device according to the PUF image



Ch6: Emerging Topics: ML and PUF Protocol

- **Challenge: CRP storage [19]**
- **A novel Solution: Machine Learning**
 - Training CRP should be compact
 - Model prediction accuracy should be high
 - Model storage size should be handlable
 - CRP transmission during mission mode should be secure



References

- [1] Randomness
- [2] Uniqueness
- [3] Diffuseness
- [4] Reliability (Steadiness)
- [5] Ruhrmair simulator
- [6] PyPUF simulator
- [7] Idriss Authentication protocol
- [8] Becker Robust FE Key Generation protocol
- [9] PUF model building attack
- [10] Becker helper data manipulation attack
- [11] PUF side channel attack
- [12] PUF replay attack
- [13] PUF fault injection attack
- [14] XOR Arbiter PUF
- [15] PUF CRP poisoning
- [16] PUF Helper Data Masking
- [17] Our secure key exchange protocol
- [18] Karimian's DLA PUF protocol
- [19] Our MDPI PUF enrollment method

Project

Implementing Memory PUF:

In this project, you will need a board with a micro chip on it which also has an on-board memory.

Example of electric boards suitable for the project:

Then you need to implement a logic code which can observe the power-up values of a designated section on the on-board memory before the firmware allocates and zeros the section.

The code should then be able to transmit the power-up memory values to a verifier system.

The verifier can be coded into a MATLAB. MATLAB supports reading through the designated universal ports on a computer. So you should define a handshake mechanism between the MATLAB verifier and the board which prepares the transmission of the PUF values from the board to the verifier.

At last step, you should collect 1000 captures of the designated section of the on-board memory and store them on the verifier system (AKA the enrollment). You should be able to do this for at least 3 boards.

Recommendation: Since 1000 captures is significant, you should develop an automated mechanism to turn-on, read, and turn-off, and turn-on the board which performs the enrollment all by itself. For this it is suggested you use relays and a separate code that handles the automation.

Appendix 1: Python simulation of Arbiter PUF

...

Appendix 2: Memory PUF on embedded boards

...

Appendix 3: Effective ANN Models for PUF cloning

...