

The Gap Between Promise and Reality: On the Insecurity of XOR Arbiter PUFs

CHES, September 16th, 2015

Georg T. Becker

Horst Görtz Institute for IT-Security, Ruhr University Bochum

hgi

Horst Görtz Institut
für IT-Sicherheit

The Promise

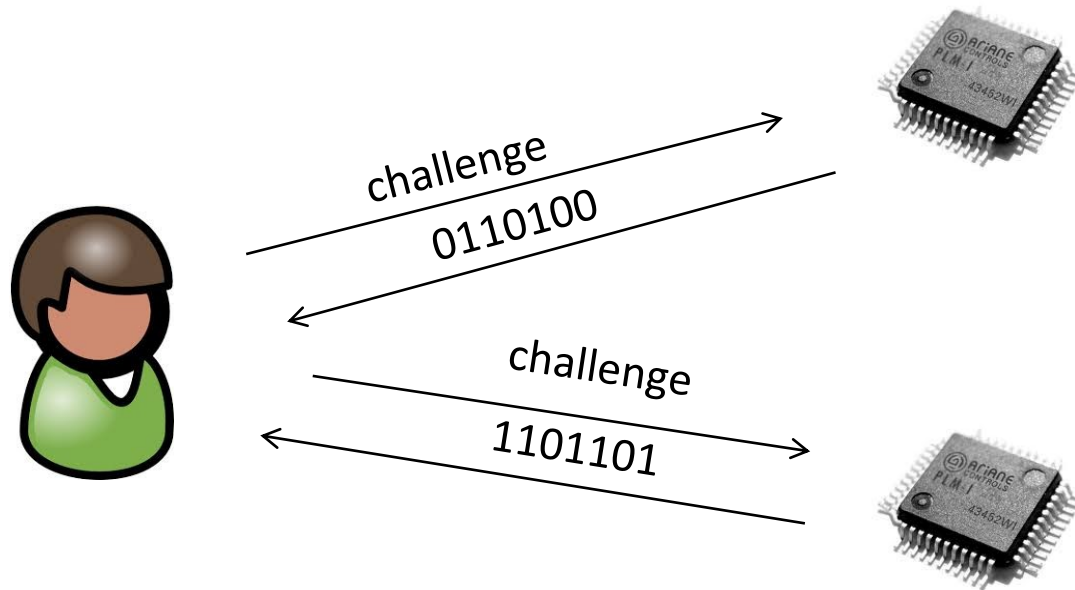
- Strong PUFs: A lightweight, secure alternative to traditional cryptography

The Gap:

- Attacking a commercial PUF based RFID tag
- New reliability based machine learning attack

Physical Unclonable Functions (PUFs)

Observation: Due to process variations, every chip has slightly different performance



⇒ Exploit this fact to give every chip a unique identity

This talk

Weak PUFs

- “Small” Challenge space
- Used for key generation and storage



Challenge 1 : 0111001
Challenge 2 : 1110100
Challenge 3 : 0100011



Challenge 1 : 0101010
Challenge 2 : 1010011
Challenge 3 : 1111010

Strong PUFs

- “Large” Challenge space
- Can be used for challenge-and-response protocols



challenge
0110100



challenge
1101101



This talk

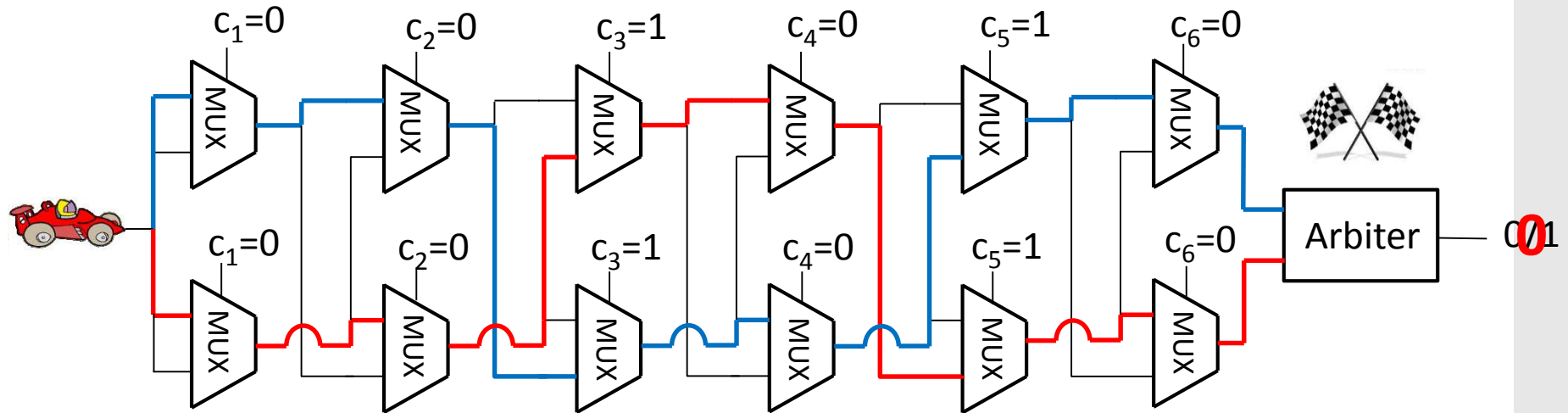
Electrical PUFs



- Can be fabricated in CMOS technology
- Example: **Arbiter PUF**, SRAM PUF, RO PUF, ...

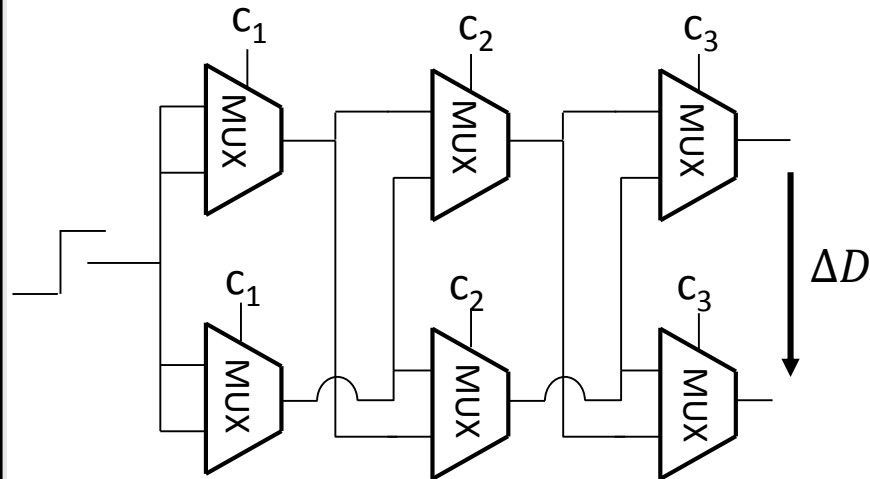
Physical PUFs

- Can not be fabricated with CMOS
- Example: Optical PUF



- Apply two race signals to delay paths with identical layouts
- A challenge defines the exact path the signals take
- Due to process variations, one signal will be faster
- Depending on which signal is faster response is 1 or 0

Software model of an Arbiter PUF



Delay *difference* determines response

$$\text{Response: } r = \begin{cases} 1 & \text{if } \Delta D > 0 \\ 0 & \text{if } \Delta D < 0 \end{cases}$$

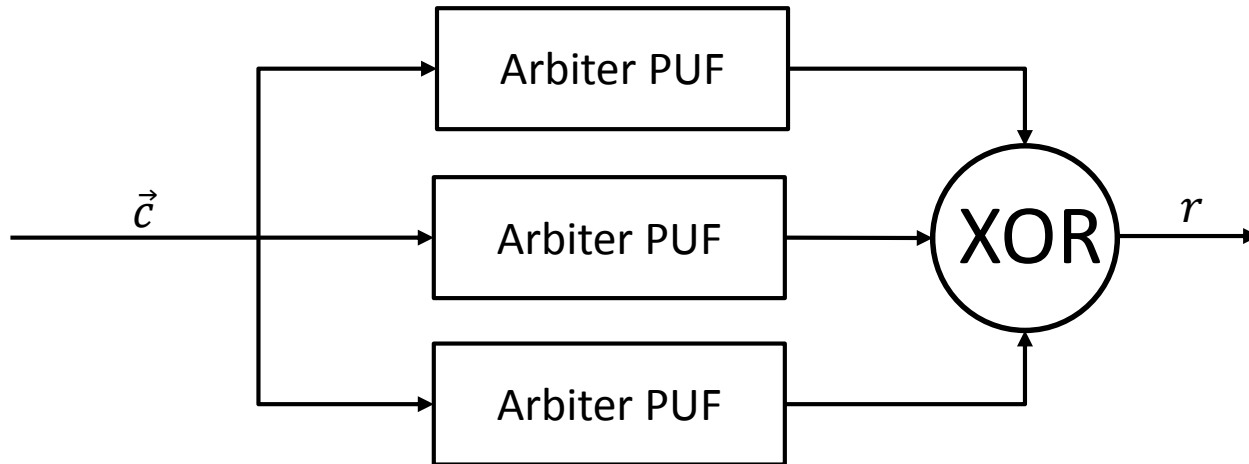
Delay difference ΔD **simply the addition** of the stage delay differences
Can be expressed as a scalar multiplication:

$$\Delta D = \vec{w} \cdot \vec{\Phi}$$

“Stage delays” with $\vec{w} \in \mathbb{R}^{N+1}$

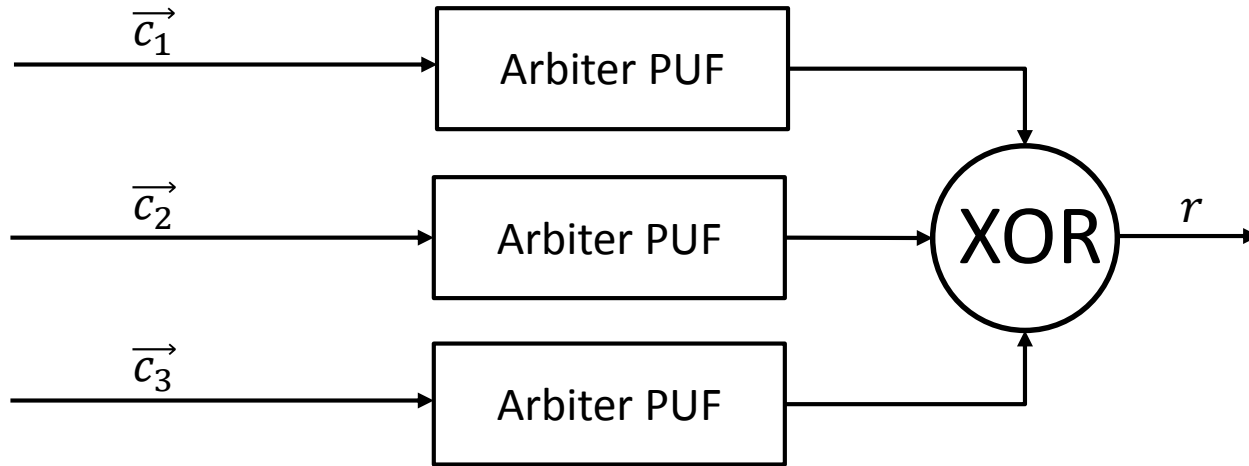
“Challenge Vector” with $\vec{\Phi} \in \{-1; 1\}^{N+1}$

XOR Arbiter PUF



Non-linearity increases attack complexity

XOR Arbiter PUF



Individual challenges further increase attack complexity

The PUF Promise:

- Secure
- Lightweight
- Unclonable – even by the manufacturer!
- No non-volatile memory needed
- Resistant against probing and reverse-engineering attacks
- Key does not need to be programmed
- More side-channel resistant (?)
- ...

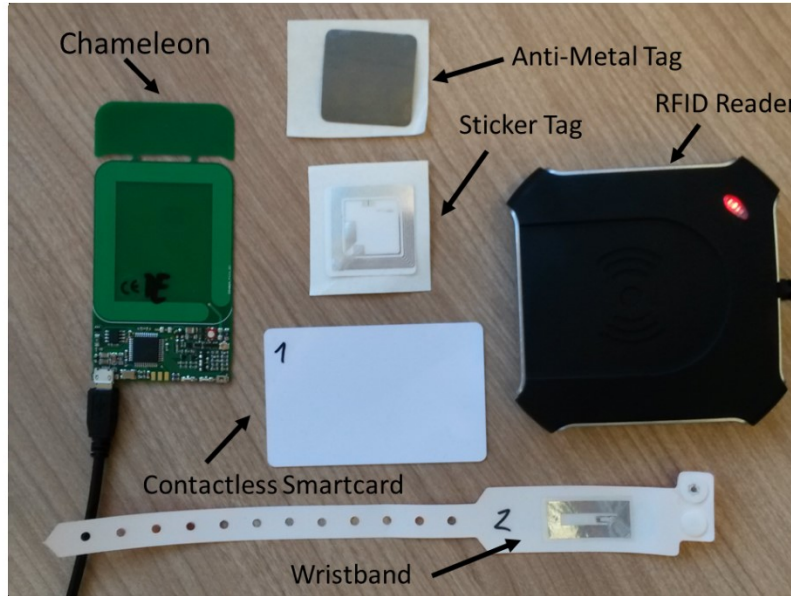
The Promise

- Strong PUFs: A lightweight, secure alternative to traditional cryptography

The Gap:

- **Attacking a commercial PUF based RFID tag**
- New reliability based machine learning attack

The Target: A PUF based commercial RFID Tag



- Available in different form factors
- Features **online** and **offline** authentication of the tags
- Costs only a few cents
- NFC compatible
- Design details not publicly available

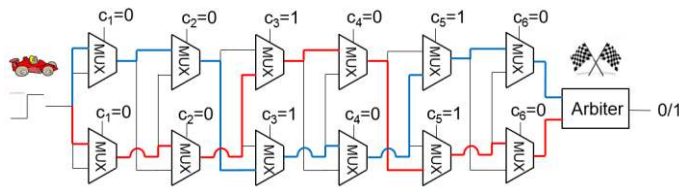
The Tags use a 4-Way PUF

64-bit LFSR (Galois LFSR)

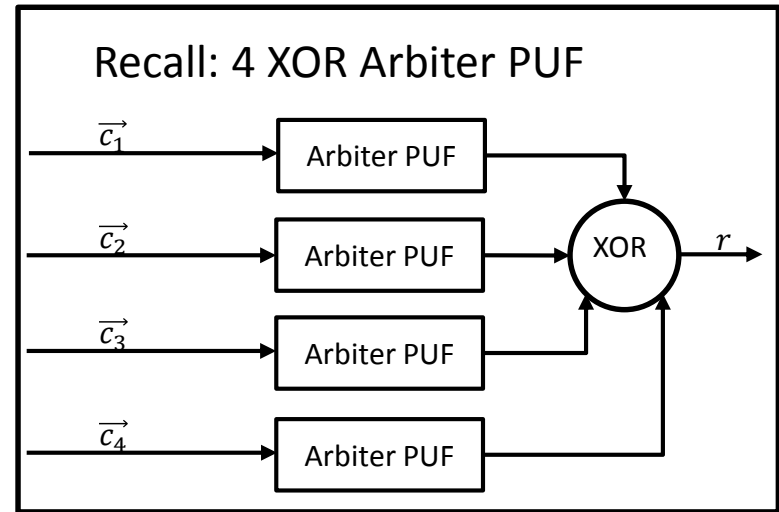
64-bit challenge

Mixer
(just permutations)

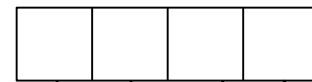
4 x 64-bit sub-challenge



64-bit Arbiter PUF



4-bit Shift register



Final output bit

Model building attack”:

- Used Logistic Regression with RPROP
- Only 1024 challenge and responses needed
- Few seconds on a Laptop
- Achieved model accuracy (85.8%) close to observed reliability 87.5%
- Measurement times only 172ms

“Cloning” of the tags:

- Build a software clone using the Chameleon
- Provided test software falsely authenticated as legitimate

⇒ PUF tags completely insecure

⇒ Real-time digital pick-pocketing possible

What if a “real” XOR Arbiter PUF would have been used?

- Used Logistic Regression with RPROP
- Only 1024 challenge and responses needed
- Measurement times of only 172ms
- Achieved model accuracy 85.8% close to the observed reliability of 87.5%
- Build a software clone using the Chameleon that was falsely authenticated as legitimate by the test software

⇒ PUF tag completely insecure

⇒ Real-time digital pick-pocketing possible

What if a “real” XOR Arbiter PUF would have been used?

The Promise

- Strong PUFs: A lightweight, secure alternative to traditional cryptography

The Gap:

- Attacking a commercial PUF based RFID tag
- **New reliability based machine learning attack**

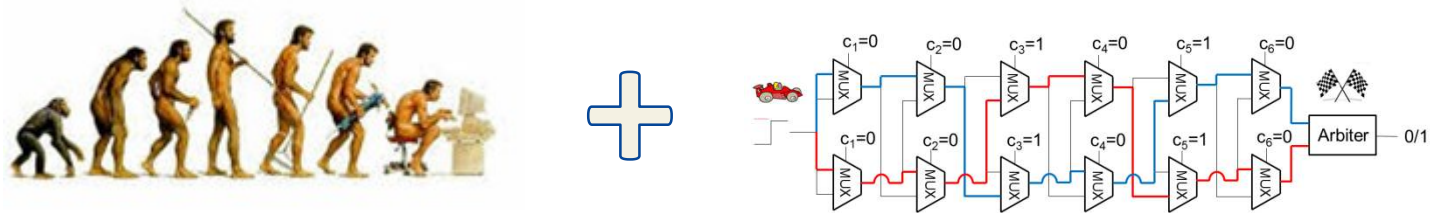
LR on XOR Arbiter PUFs

Stages	XORs	CRPs	Convergence rate	CRP increase
64	4	12,000	0.29 (58/200)	-
64	5	90,000	0.28 (56/200)	7.5
64	6	750,000	0.26 (52/200)	8.3
64	7	5,000,000	0.31 (10/32)	6.7
64	8	50,000,000	0.50 (14/28)	10
64	9	350,000,000	0.25 (2/8)	7
128	4	65,000	0.26 (52/200)	-
128	5	975,000	0.26 (52/200)	15
128	6	22,000,000	0.25 (2/8)	22.6
128	7	400,000,000	0.38 (2/8)	18.2

Machine Learning Complexity *increases exponentially* with the number of XORs

J. Tobisch and G.T. Becker “On the Scaling of Machine Learning Attacks on XOR Arbiter PUFs with Application to Noise Bifurcation”, RFIDSec 2015

Attack based on Ruhrmair et. al. “Modelling Attacks on PUFs” ACM CCS 2010



Start:

1. Create a **parent** by setting the delay vector \vec{w} to all zeros

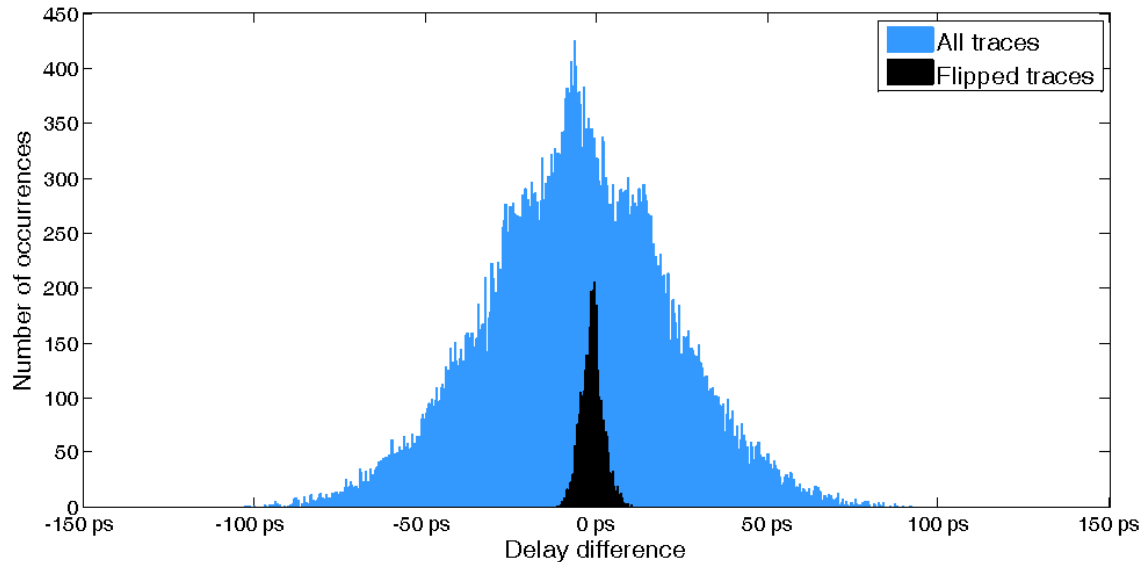
Repeat:

1. Generate **children** by randomly modifying the parent's parameters (delay vector)
2. Test the **fitness** of these children
3. Keep the fittest children as **parents** for **next generation**

⇒ The PUF models gradually become more and more accurate

How do we determine the Fitness of a PUF model?

Reliability of Arbiter PUFs

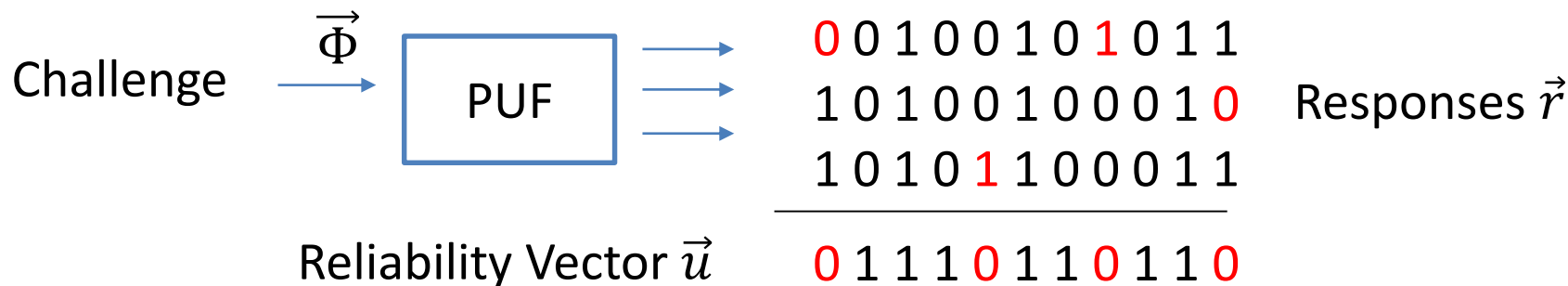


Analysis of which challenges flip when the voltage is increased and decreased by 0.1V

The closer the delay difference to zero, the more likely the response is unreliable
⇒ The information which challenges flipped can be used to model the PUF

Fitness function based on reliability

Setup: Measure responses 3 times to get a reliability vector h



Given: PUF model \vec{w} , challenge matrix $\vec{\Phi}$ and reliability vector \vec{u}

1) Compute a hypothetical reliability vector $\vec{h} = h_1, \dots, h_n$:

$$\Delta D_i = \vec{w} \cdot \vec{\Phi}_i \quad h_i = \begin{cases} 1 & \text{if } |\Delta D_i| > \epsilon \\ 0 & \text{if } |\Delta D_i| \leq \epsilon \end{cases}$$

2) Compute the correlation coefficient between \vec{u} and \vec{h}

⇒ The higher the correlation coefficient, the fitter the PUF model

Key observation:

If one of the Arbiter PUFs is unreliable for a given challenge, the final response of the XOR PUF is also unreliable

Given:

- The reliability vector u_i of one Arbiter PUF
- The reliability vector u_{xor} of the entire XOR Arbiter PUF

Then there is a **linear relationship between u_{xor} and u_i**

⇒ Correlation coefficient: $\text{corrcoef}(\overrightarrow{u_{xor}}, \overrightarrow{u_i}) > 0$

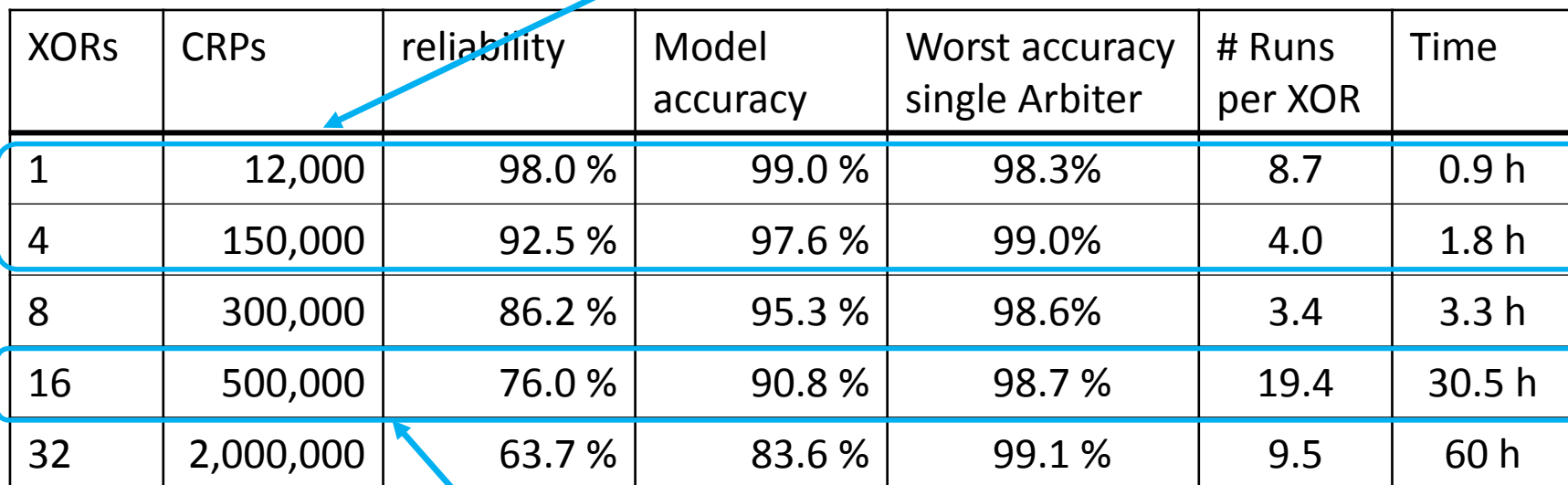
The higher the correlation coefficient between the measured reliability vector $\overrightarrow{u_{xor}}$ and a hypothetical reliability $\overrightarrow{h_i}$, the more accurate is this PUF model!

So, why is this reliability based fitness function cool?

- We can use a **divide-and-conquer approach**
 - ⇒ We attack one Arbiter PUF at a time
 - Each additional XOR is only seen as an increase in noise during one machine learning run
 - The relative noise added by an additional XOR decreases with the number of XORs
- ⇒ The attack complexity (number of needed CRPS) only **increases linearly** with the number of XORs!

- Attack results of a reliability-based machine learning attack on a simulated 128-Stage XOR Arbiter PUF, with different challenges for each Arbiter.
- Time refers to the average attack time using 16 cores of a 64 core cluster (while all cores were active).

Worse than state-of-the-art machine learning attacks



XORs	CRPs	reliability	Model accuracy	Worst accuracy single Arbiter	# Runs per XOR	Time
1	12,000	98.0 %	99.0 %	98.3%	8.7	0.9 h
4	150,000	92.5 %	97.6 %	99.0%	4.0	1.8 h
8	300,000	86.2 %	95.3 %	98.6%	3.4	3.3 h
16	500,000	76.0 %	90.8 %	98.7 %	19.4	30.5 h
32	2,000,000	63.7 %	83.6 %	99.1 %	9.5	60 h

Somewhere in the area of 15^{11} times more efficient than state-of-the-art machine learning attacks

Results using data from the commercial PUF based RFID Tags

- To show that this attack also works in practice the commercial PUF tags are used
- The output of several tags are XORed to build a n-XOR-4Way PUF (with 64 stage Arbiter PUFs)

Number of PUFs	Total XORs	CRPs	reliability	Model accuracy	Accuracy 4-Way PUF	Time
1	4	4,000	87.5 %	87.1%	87.1 %	0.7 m
2	8	10,000	80.0 %	78.5 %	88.0%	1.6 m
4	16	40,000	69.2 %	67.2 %	87.9%	3.3 m
8	32	400,000	56.3 %	55.6 %	87.5 %	13.1 m

⇒ **Bottom Line:** Attack also possible with real measurement data

- Analyzed commercial PUF tags.
- Can be attacked in 172 ms
- ⇒ Very far from being secure, mainly security by obscurity

- New Reliability based Machine Learning Attack on XOR Arbiter PUFs
- Attack **uses a divide-and-conquer strategy**
- Attack complexity increases only linearly with the number of XORs
- ⇒ **XOR Arbiter PUFs insecure** regardless of the number of XORs

- Results not limited to XOR PUFs. See for example [TCAD15] for an attack on the Reverse-Fuzzy Extractor protocol

Currently, electrical strong PUFs very far away
from being secure

Thank you very much!
Any questions?

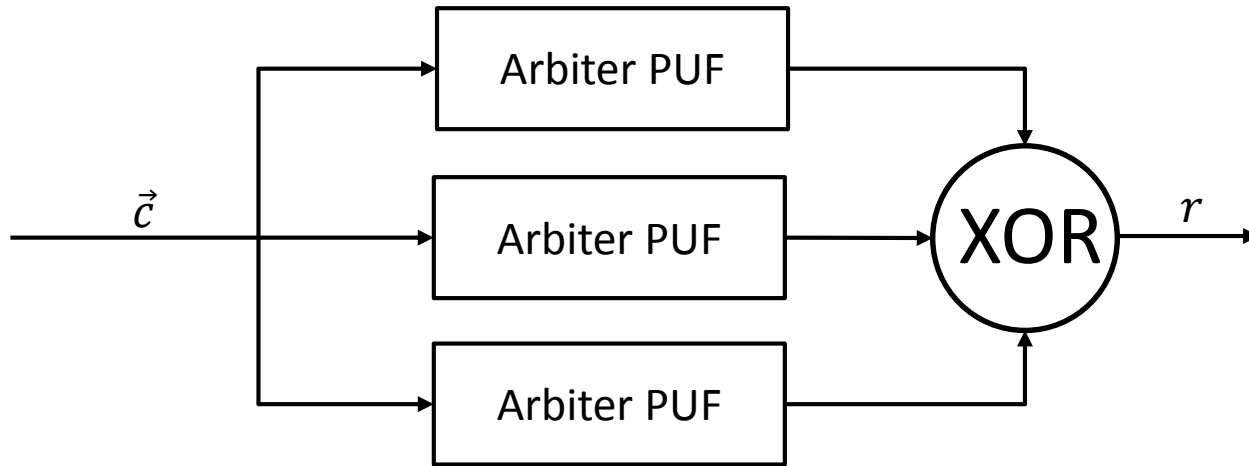
Backup Slides



Results of the reliability based attacks for different noise levels

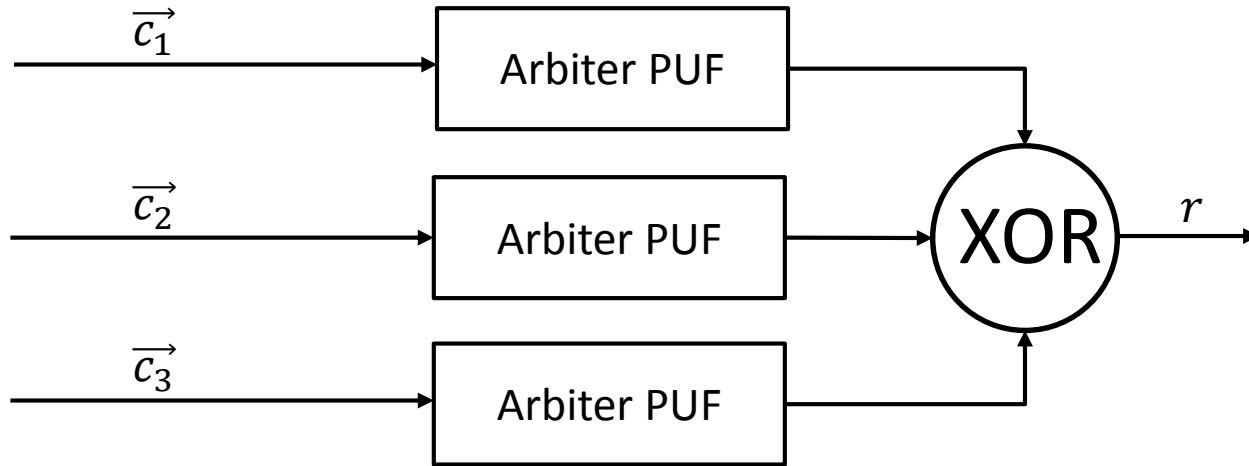
Table 3. Results of reliability-based CMA-ES attacks on a 8 XOR, 128 stages PUF with different noise values σ_{noise} . Again 10 attacks were performed per entry.

σ_{noise}	Reliability single Arbiter	Reliability XOR PUF	# CRPs	Accuracy reference set	#runs per XOR	Accuracy single Arbiter	Time
0.1	99.8 %	98.4 %	$2500 \cdot 10^3$	96.0 %	6.8	98.9 %–99.7 %	16.7 h
0.25	99.5 %	96.2 %	$1000 \cdot 10^3$	95.5 %	8.1	97.2 %–99.8 %	17.0 h
0.5	99.0 %	92.6 %	$500 \cdot 10^3$	94.7 %	7.5	98.9 %–99.8 %	6.6 h
1	98.0 %	86.2 %	$300 \cdot 10^3$	95.3 %	3.4	98.6 %–99.7 %	3.3 h
2	96.0 %	75.8 %	$200 \cdot 10^3$	94.5 %	1.6	98.8 %–99.6 %	1.2 h
4	92.1 %	62.7 %	$100 \cdot 10^3$	84.6 %	8.5	96.2 %–98.2 %	4.6 h



$$\text{Best Model: } \Delta D = \prod_{i=1}^l \vec{w}_i^T * \vec{\Phi}$$

Non-linearity increases attack complexity



$$\text{Best Model: } \Delta D = \prod_{i=1}^l \vec{w}_i^T * \vec{\Phi}_i$$

Individual challenges further increase attack complexity

Results

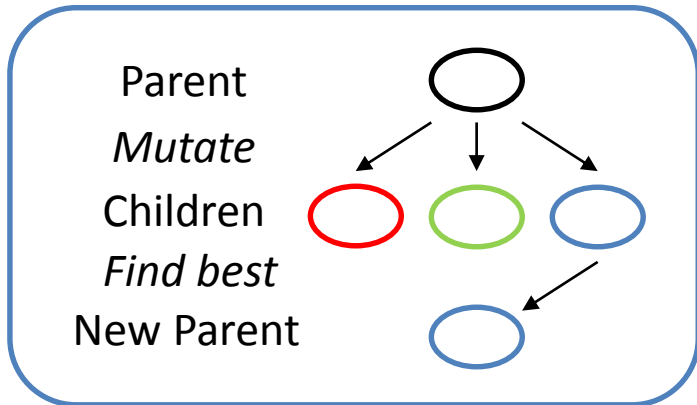
- Attack results of a reliability-based machine learning attack on a simulated 128-Stage XOR Arbiter PUF, with different challenges for each Arbiter.
- Attack performed on a cluster with 4 nodes, 64 cores each. 16 cores are used in each attack and attacks are executed in parallel.

XORs	CRPs	Reliability	Model accuracy	Worst accuracy single Arbiter	# Runs per XOR	Time
1	12,000	98.0 %	99.0 %	98.3%	8.7	0.9 h
4	150,000	92.5 %	97.6 %	99.0%	4.0	1.8 h
8	300,000	86.2 %	95.3 %	98.6%	3.4	3.3 h
16	500,000	76.0 %	90.8 %	98.7 %	19.4	30.5 h
32	2,000,000	63.7 %	83.6 %	99.1 %	9.5	60 h
4	150,000	92.5 %	97.7 %	99.1 %	4.2	1.1 h
8	300,000	86.2 %	95.7 %	99.1 %	7.2	3.3 h
16	500,000	76.1 %	90.0 %	98.7 %	30.6	34 h

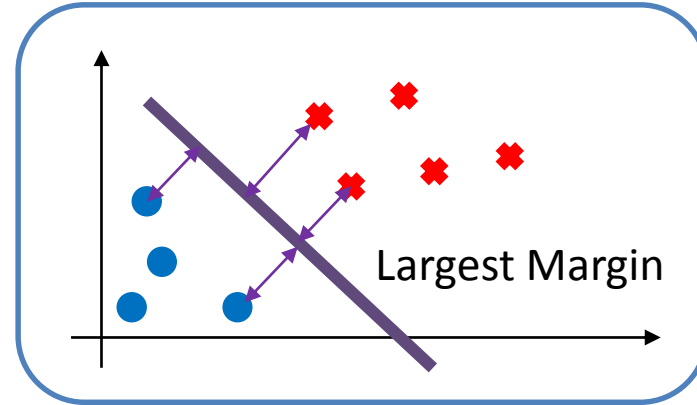
Bottom:

Results if the same challenge is used. Note that for 16 XORs a 2-step approach was used

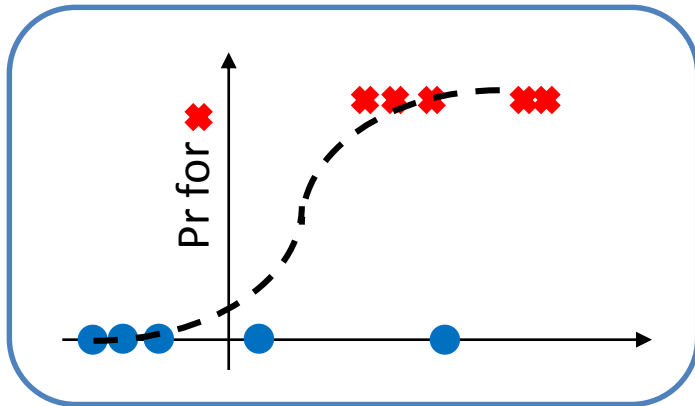
Evolution Strategy (ES)



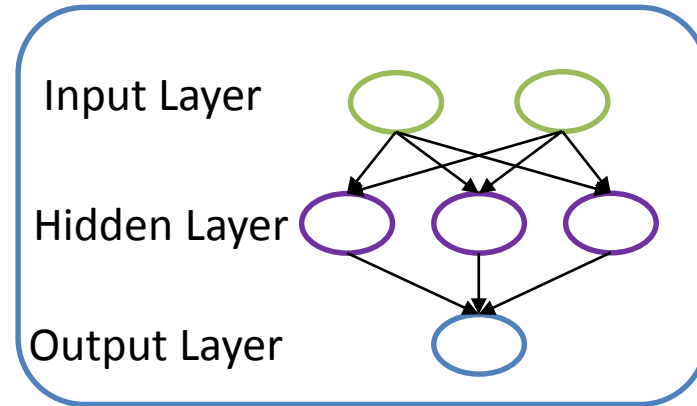
Support Vector Machine (SVM)



Logistic Regression



Artificial Neural Networks (ANN)



Online Authentication

Initialization Phase:

- The verifier collects random challenge and responses and stores them in a data base

Authentication Phase:

- Verifier sends one if the stored challenges to the tag
- If the tag's PUF response matches the response stored in the data base, tag is authenticated

Offline Authentication

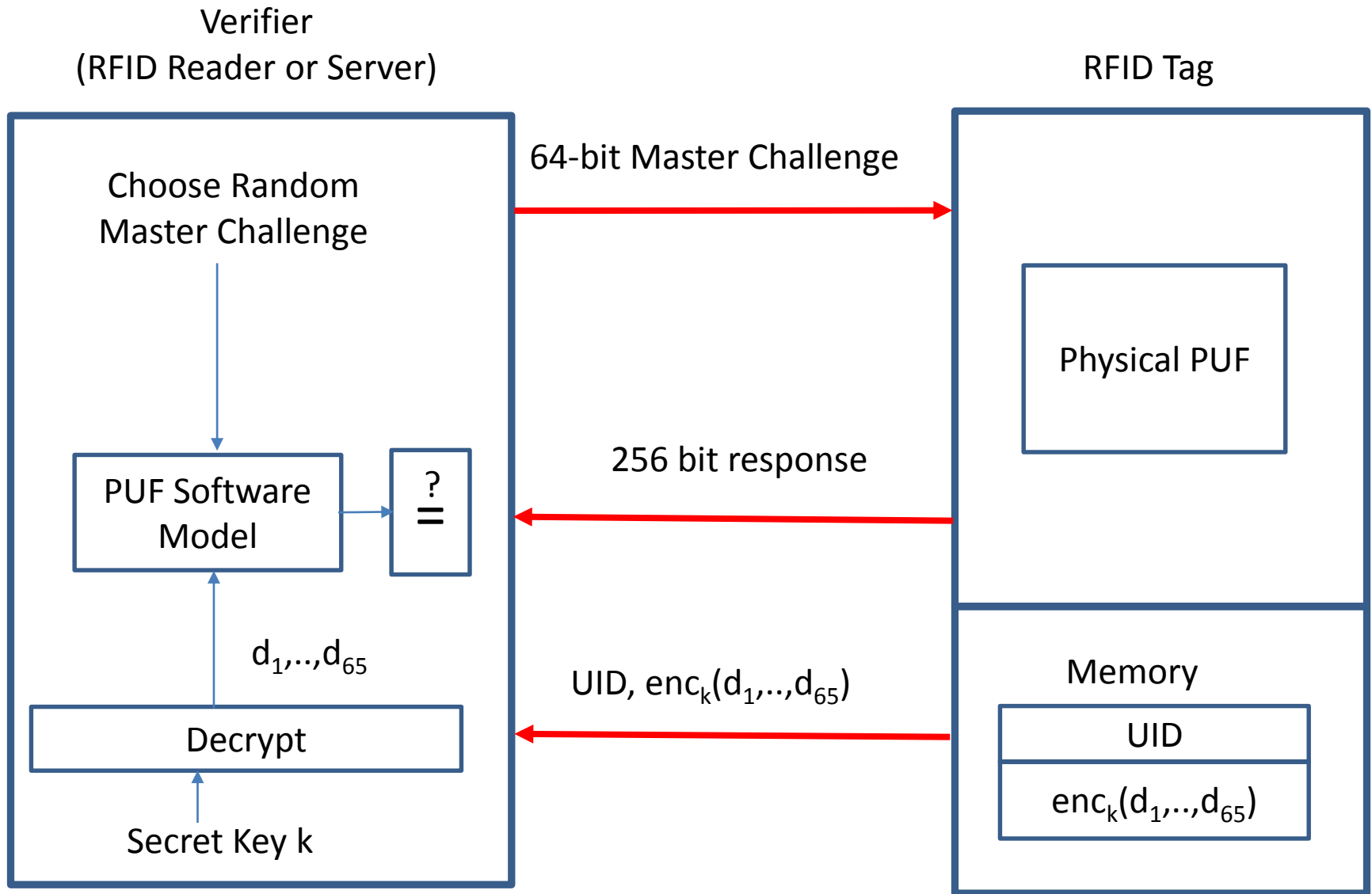
Initialization Phase:

- Internal delay difference of the PUF are determined [by directly revealing the PUF responses without any XOR]
- These delay differences are encrypted by the verifier and stored in the (public) memory of the tag

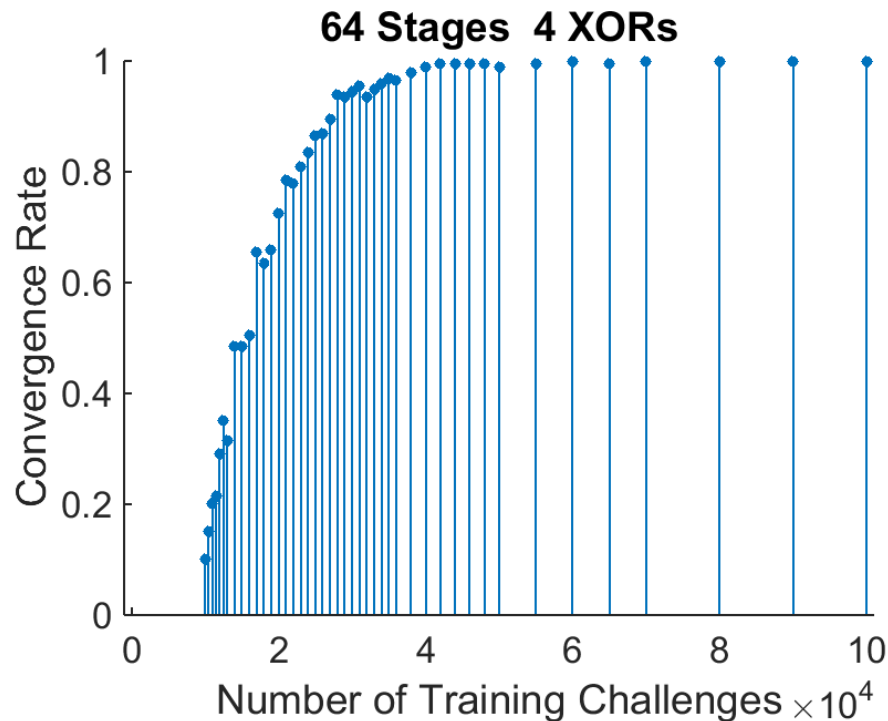
Authentication Phase:

- Next slide

Offline PUF Authentication

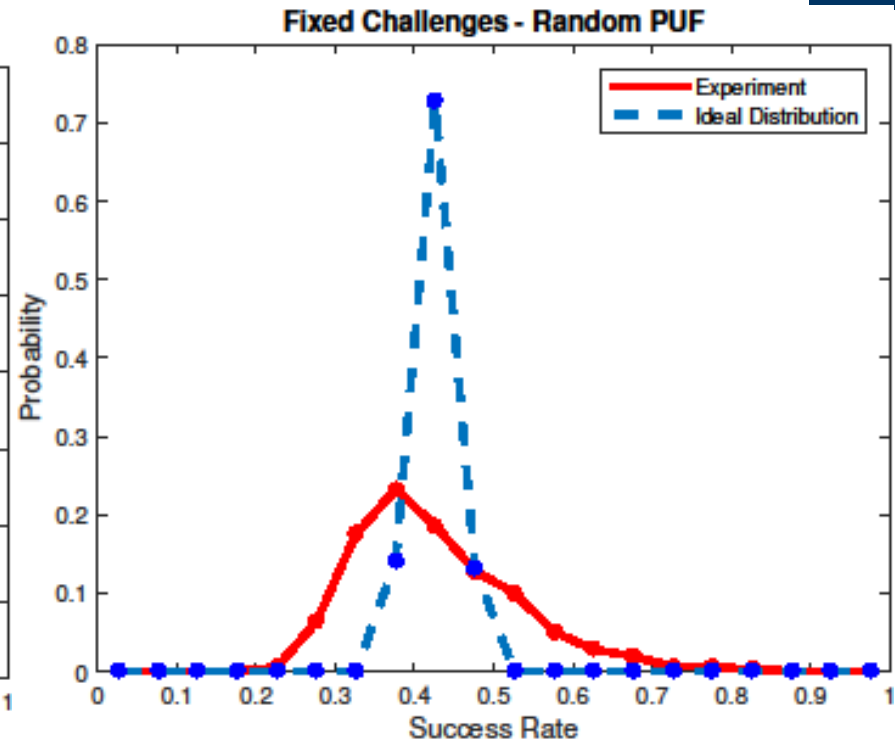
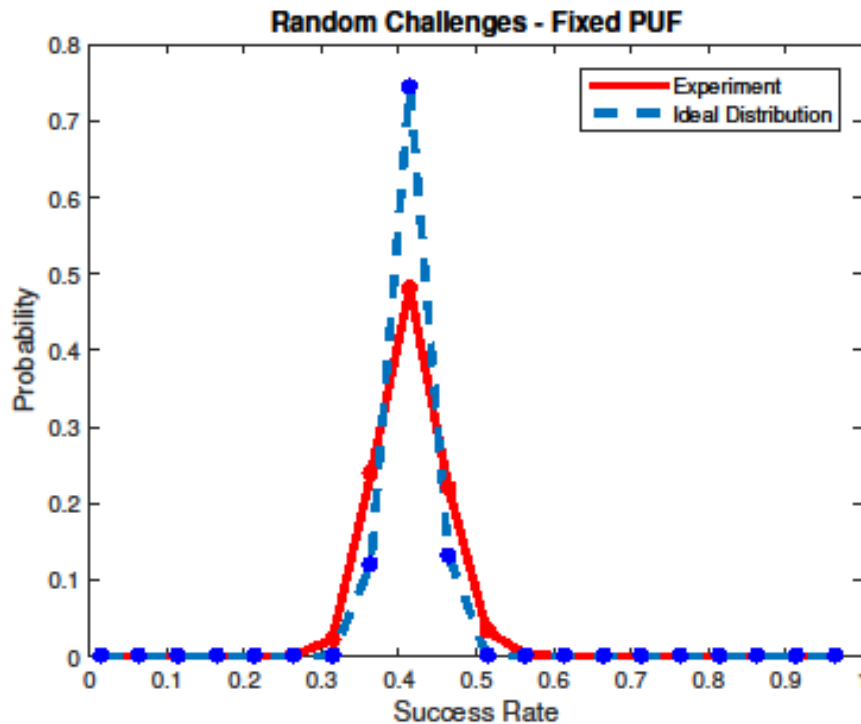


Machine Learning Attack on 4-XOR Arbiter PUF



- Reducing the number of challenges reduces the convergence rate
- But how low of a convergence rate is acceptable? How many challenges are “required”?

Machine Learning Attack on 4-XOR PUF



Result of this experiment:

- Some instances are easier to attack than others!
- Should we care about “average” attack complexity or “best case” attack complexity?
- How do we find the PUF instance that has the lowest attack complexity?

Observation:

A 4-XOR PUF response is 100% reliable if, and only if, all of the 4 sub-responses are also 100% reliable

The XOR reliability vector can be expressed as the product of the individual reliability vectors

$$u_{xor} = u_1 \cdot u_2 \cdot u_3 \cdot u_4 \text{ with } u_i \in \{0; 1\}$$

The probability that $u_{xor} \neq u_1$ can be expressed as:

$$P(u_{xor} \neq u_1) = P(u_1) \cdot P(u_2 = 0 | u_3 = 0 | u_4 = 0) \text{ with } u_i \in \{0; 1\}$$

Hence, there is a **linear relationship between u_{xor} and u_i**

⇒ Correlation coefficient: $\text{corrcoef}(\vec{u}_{xor}, \vec{u}_i) > 0$

The higher the correlation coefficient between the measured reliability vector \vec{u}_{xor} and a hypothetical reliability \vec{h}_i , the more accurate is this PUF model!

Step 1:

- Reverse-Engineer PUF design
- (only software reverse-engineering necessary)

Step 2:

- Perform Machine Learning attack to recover delay values

Step 3:

- Make a software clone using the smartcard emulated Chameleon