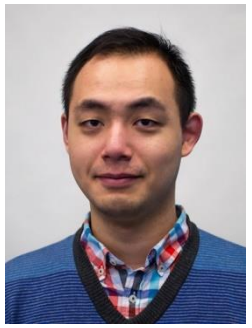
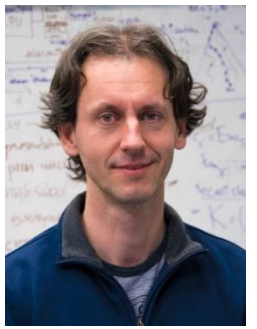
The Interpose PUF: Secure PUF Design against State-of-the-art Machine Learning Attacks

Phuong Ha Nguyen,
Durga P. Sahoo, Kaleel Mahmood, Chenglu Jin,
Ulrich Rührmair and Marten van Dijk





Ha Durga Chenglu Kaleel Uli Marten

CHES 2019

UCONN

UCONN

Content

1. Concept - Overview - Motivation
2. APUF- XOR APUF -iPUF
3. Machine Learning based Modeling Attacks on PUFs
4. Classical Machine Learning Attacks on APUFs
5. Classical Machine Learning Attacks on XOR APUFs
6. Interpose PUF (iPUF) – Resistance against Classical Machine Learning Attacks
7. Short-term Reliability
8. Reliability based Modeling Attacks on APUFs, XOR APUFs and iPUF's resistance against reliability based modeling attacks.

UCONN

1. Concept - Overview - Motivation

UCONN

Concept - Overview – Motivation

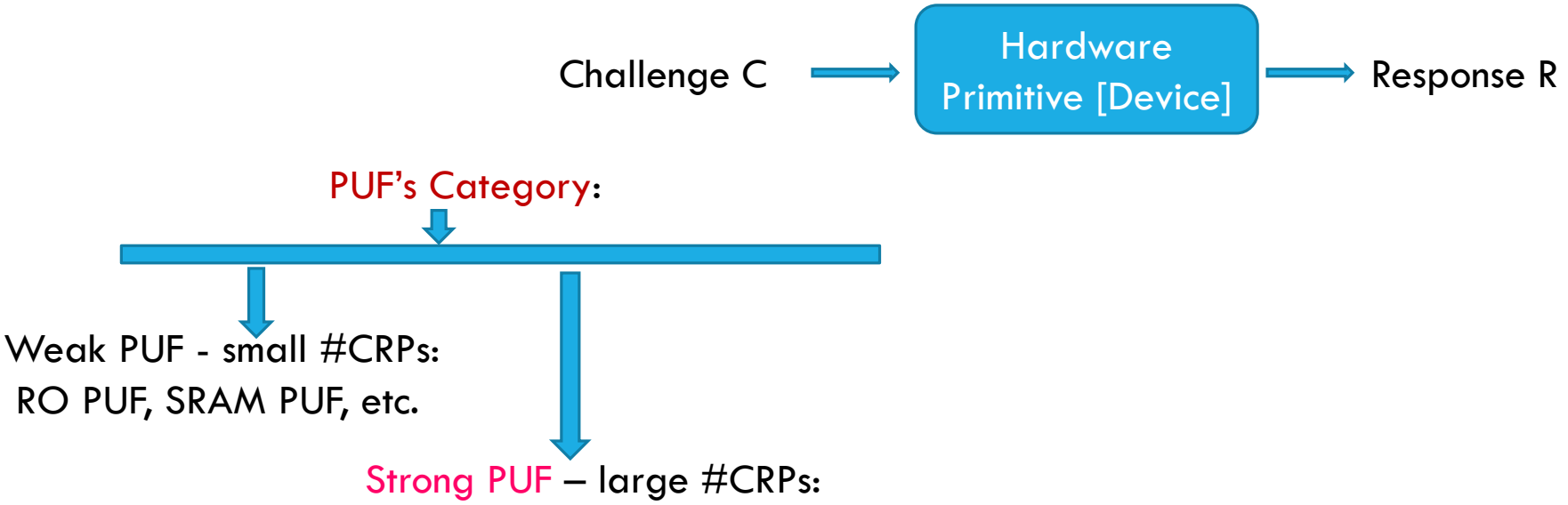
Concept - Overview – Motivation

Nature: process variation – physically unclonability - unique

Application: device identification, authentication and crypto key generation

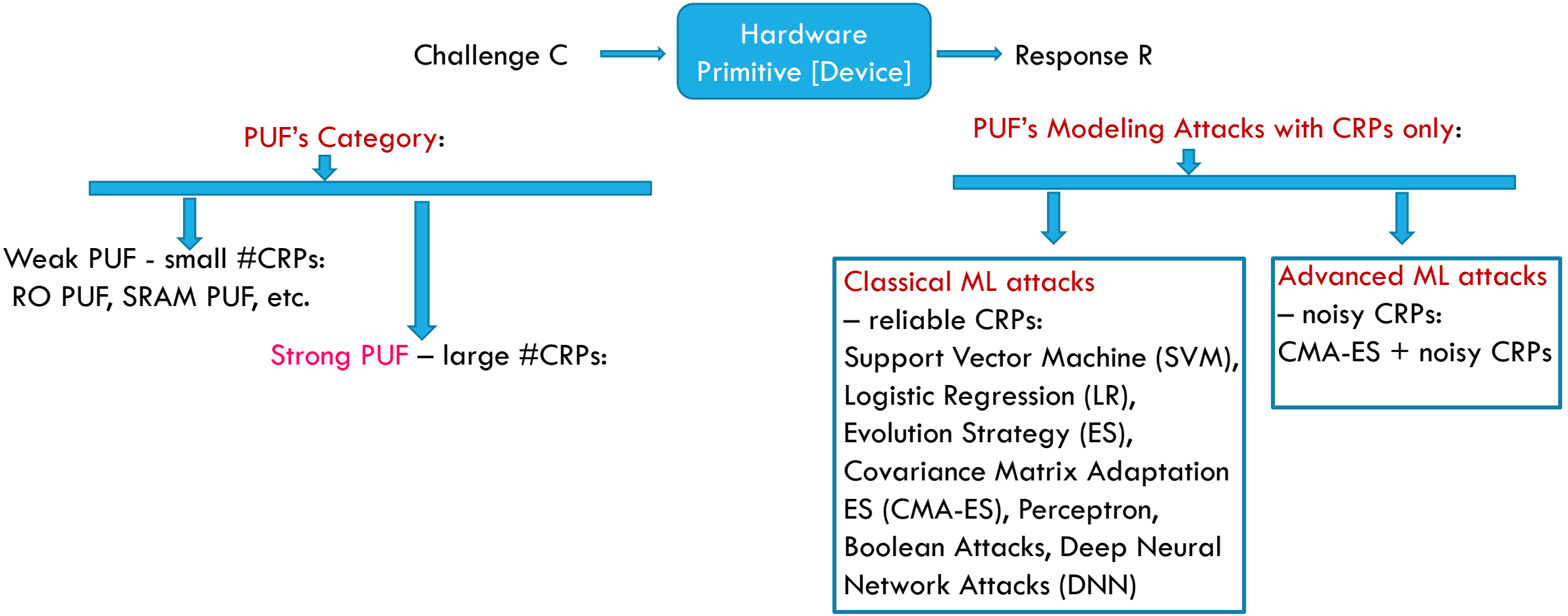
UCONN

Concept - Overview – Motivation



UCONN

Concept - Overview – Motivation



UCONN

Concept - Overview – Motivation

Challenge C → Hardware Primitive [Device] → Response R

PUF's Category:

Weak PUF - small #CRPs:
RO PUF, SRAM PUF, etc.

Strong PUF – large #CRPs:

Broken but lightweight:
APUF, XOR APUF, Feed Forward PUF, Lightweight Secure PUF, Bistable Ring PUF, MPUF

No Security Proof:
Power Grid PUF, Clock PUF, Crossbard PUF

Security Proof:
LPN PUFs
- heavy

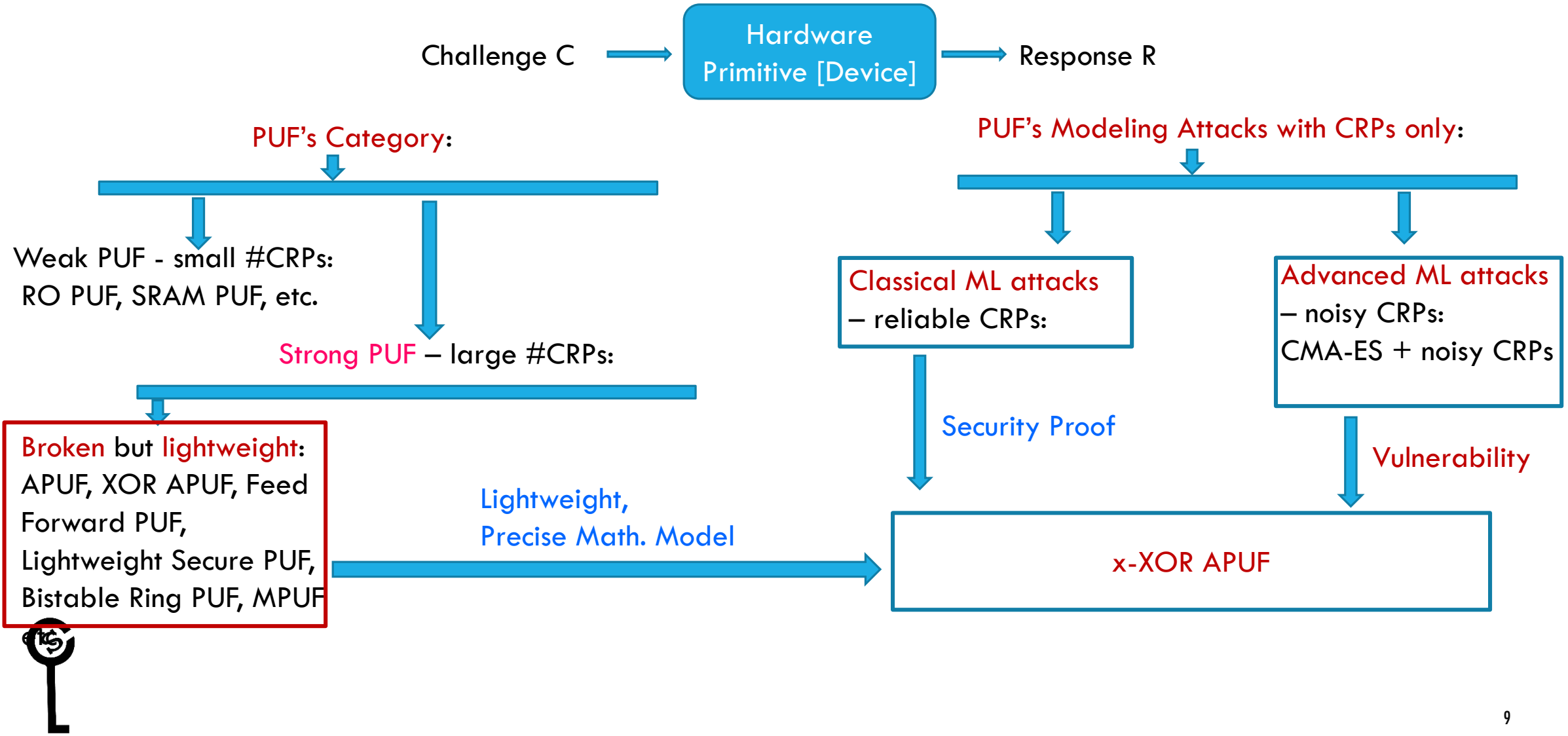
PUF's Modeling Attacks with CRPs only:

Classical ML attacks
– reliable CRPs:
Support Vector Machine (SVM), Logistic Regression (LR), Evolution Strategy (ES), Covariance Matrix Adaptation ES (CMA-ES), Perceptron, Boolean Attacks, Deep Neural Network Attacks (DNN)

Advanced ML attacks
– noisy CRPs:
CMA-ES + noisy CRPs

UCONN

Concept - Overview – Motivation



UCONN

Concept - Overview – Motivation

Challenge C → Hardware Primitive [Device] → Response R

PUF's Category:

Weak PUF - small #CRPs:
RO PUF, SRAM PUF, etc.

Strong PUF – large #CRPs:

Broken but lightweight:
APUF, XOR APUF, Feed Forward PUF, Lightweight Secure PUF, Bistable Ring PUF, MPUF etc.

Lightweight, Precise Math. Model

PUF's Modeling Attacks with CRPs only:

Classical ML attacks – reliable CRPs:

Security Proof

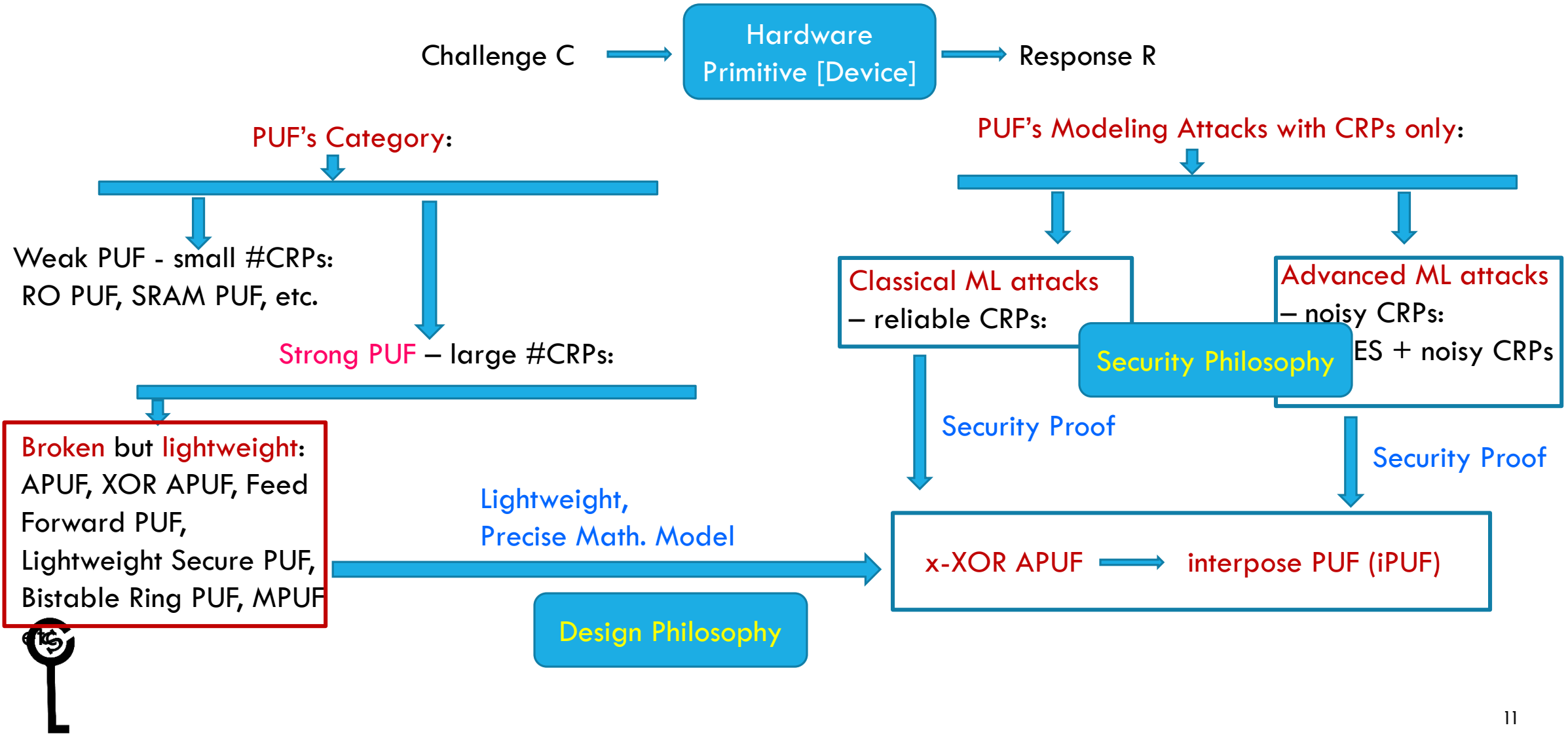
Advanced ML attacks – noisy CRPs:
CMA-ES + noisy CRPs

Security Proof

x-XOR APUF → interpose PUF (iPUF)

UCONN

Concept - Overview – Motivation

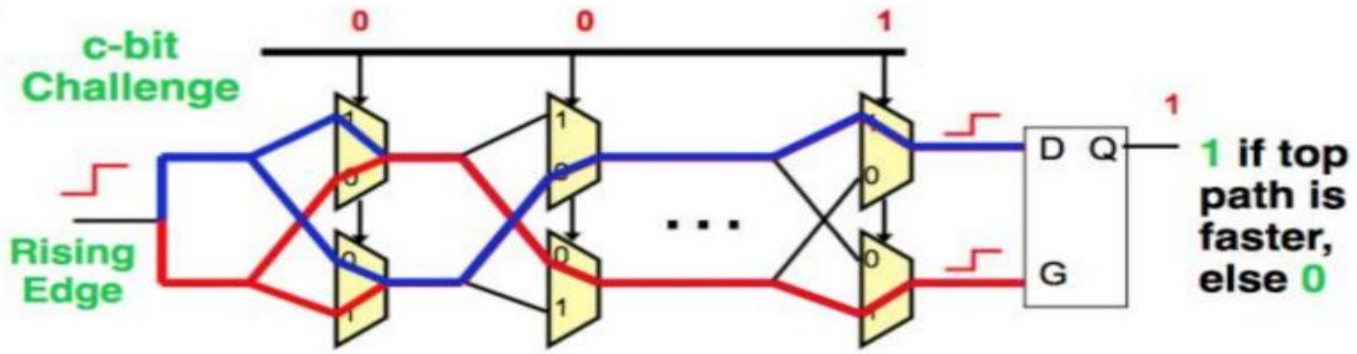


UCONN

2. APUF- XOR APUF -iPUF

UCONN

APUF, XOR APUF and iPUF



Arbiter PUF (APUF) [1]

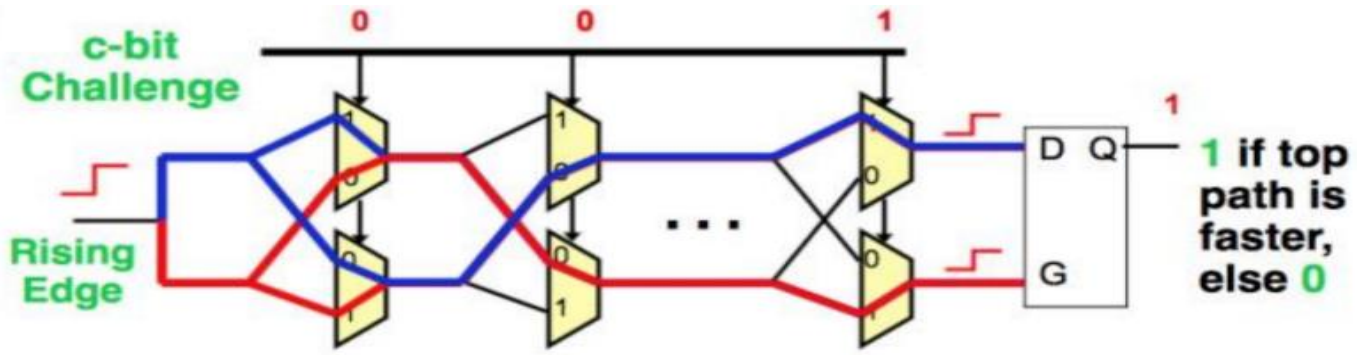
- Extremely lightweight and large number of CRPs i.e, 2^n CRPs

- Process variation makes PUF's output not always reliable

- Not secure against modeling attacks

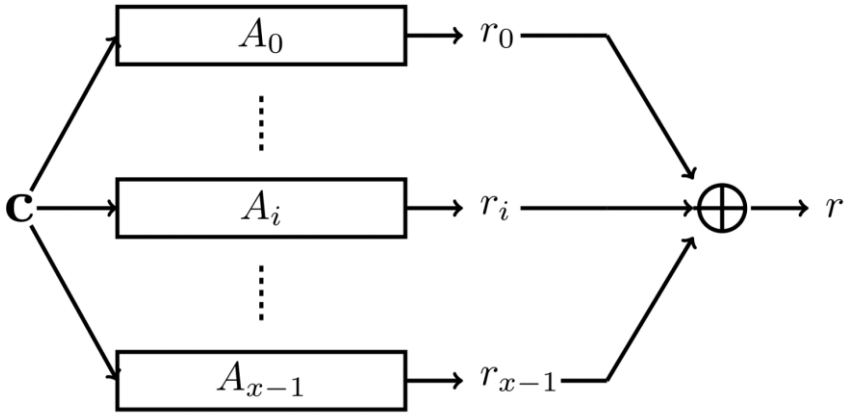
UCONN

APUF, XOR APUF and iPUF



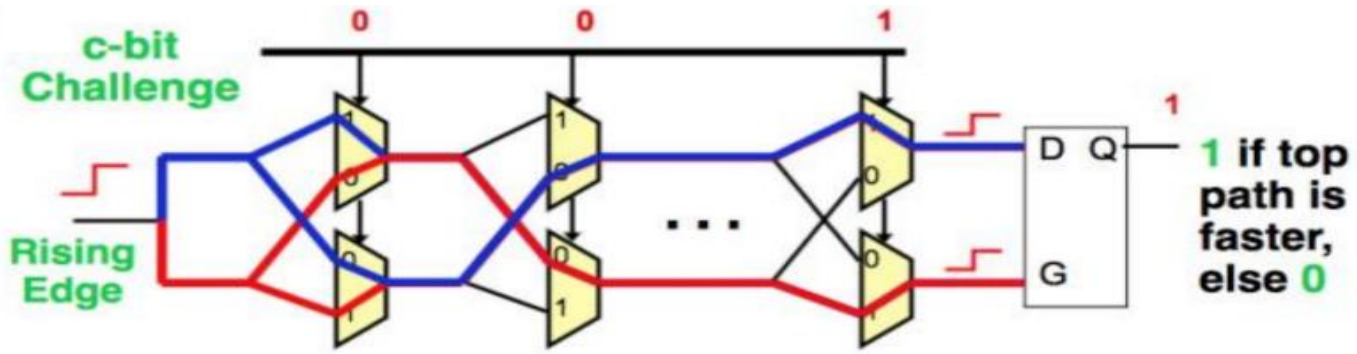
Arbiter PUF (APUF)

x-XOR APUF



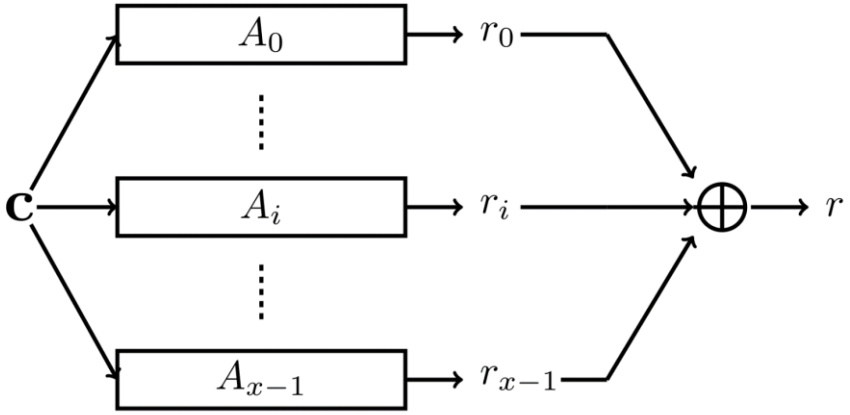
UCONN

APUF, XOR APUF and iPUF

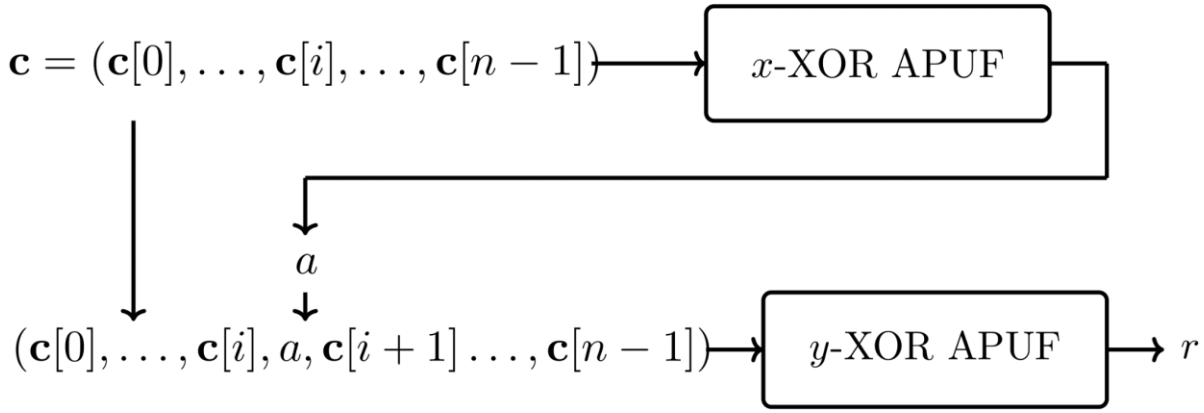


Arbiter PUF (APUF)

x-XOR APUF



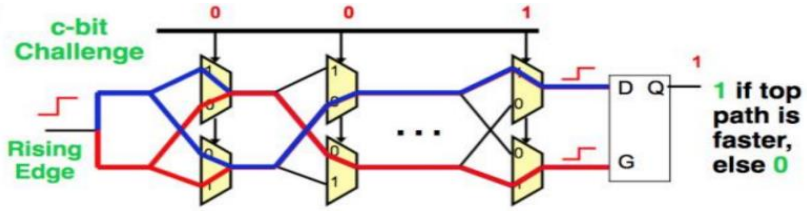
Interpose PUF (iPUF)

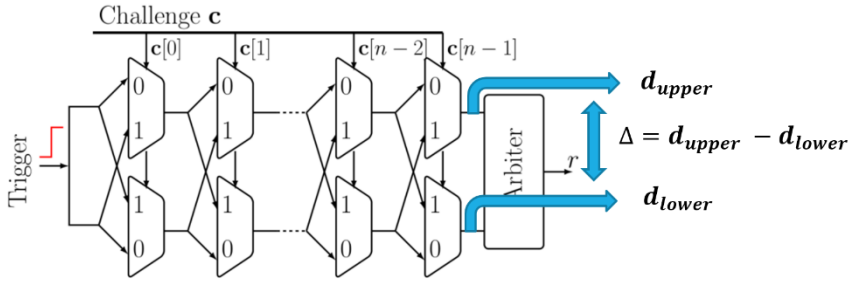


UCONN

APUF, XOR APUF and iPUF

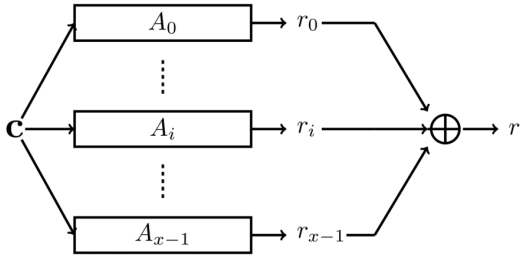
Arbiter PUF (APUF)





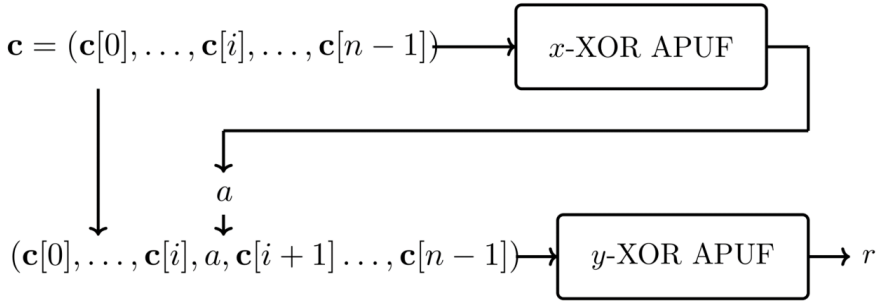
A n-stage classic Arbiter PUF with challenge $\mathbf{c} \in \{0,1\}^n$.

x-XOR Arbiter PUF



x-XOR PUF.

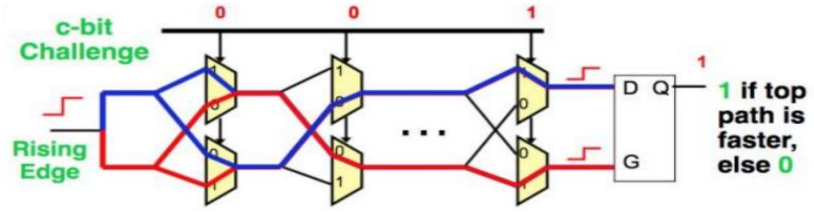
Interpose PUF (iPUF)

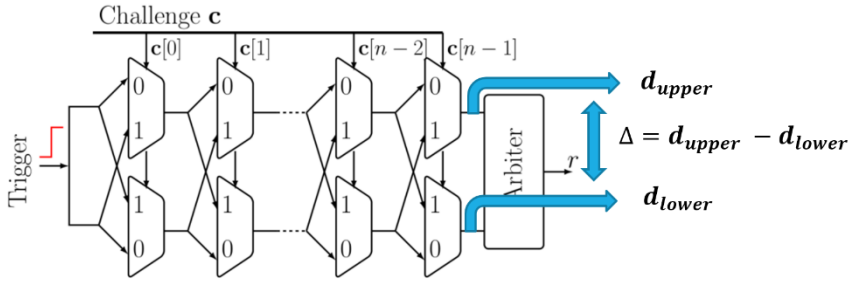


UCONN

APUF, XOR APUF and iPUF

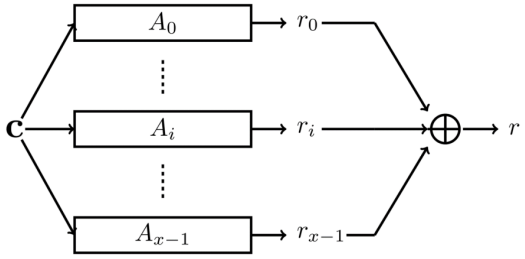
Arbiter PUF (APUF)





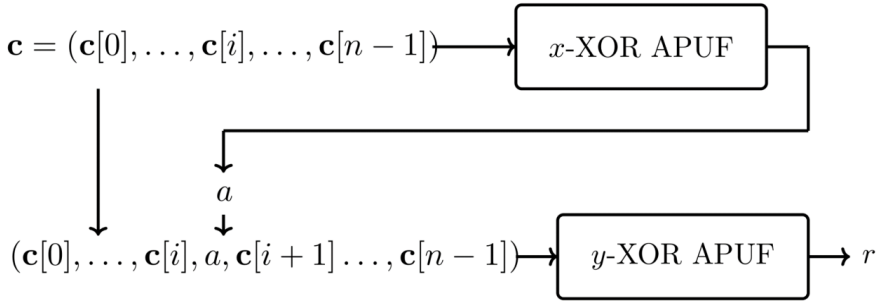
A n-stage classic Arbiter PUF with challenge $\mathbf{c} \in \{0,1\}^n$.

x-XOR Arbiter PUF



x-XOR PUF.

Interpose PUF (iPUF)



- $\Delta > 0 \rightarrow r = 1. Otherwise\ r = 0$
- $\Delta = \boldsymbol{d_{upper}} - \boldsymbol{d_{lower}} = \boldsymbol{w} \cdot \boldsymbol{\Phi}$
- $\boldsymbol{w}: unique\ for\ any\ APUF\ instance$
- $\boldsymbol{\Phi}\ is\ the\ parity\ vector$

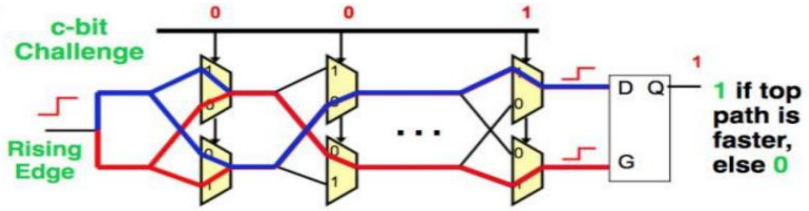
$\boldsymbol{\Phi}[i] = \prod_{j=i,\ldots,n-1}(1 - \boldsymbol{c}[j]), i = 0, \ldots, n-1\ ,\ \boldsymbol{\Phi}[n] = 1$

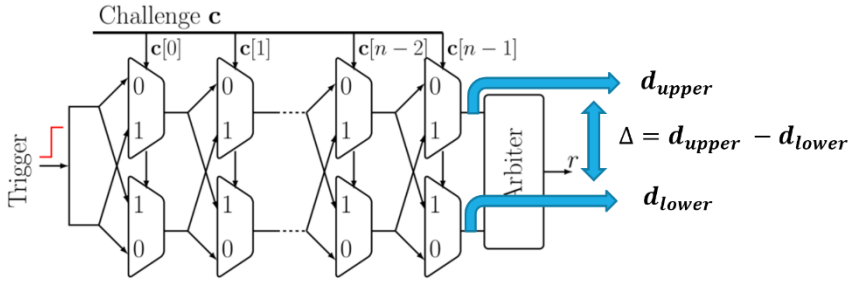
Precise linear model + CRPs + ML
= practically and softwarelly clonable

UCONN

APUF, XOR APUF and iPUF

Arbiter PUF (APUF)





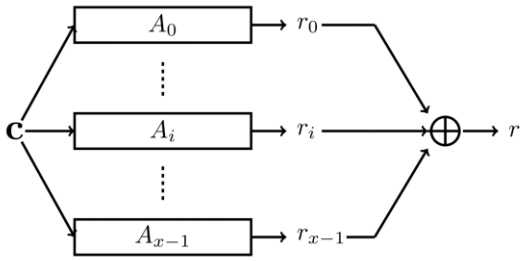
A n-stage classic Arbiter PUF with challenge $\mathbf{c} \in \{0,1\}^n$.

- $\Delta > 0 \rightarrow r = 1. Otherwise\ r = 0$
- $\Delta = \boldsymbol{d_{upper}} - \boldsymbol{d_{lower}} = \boldsymbol{w} \cdot \boldsymbol{\Phi}$
- $\boldsymbol{w}$: $unique\ for\ any\ APUF\ instance$
- $\boldsymbol{\Phi}\ is\ the\ parity\ vector$

$\boldsymbol{\Phi}[i] = \prod_{j=i,\dots,n-1}(1 - \boldsymbol{c}[j]), i = 0, \dots, n-1\ ,\ \boldsymbol{\Phi}[n] = 1$

Precise linear model + CRPs + ML
= practically and softwarelly clonable

x-XOR Arbiter PUF

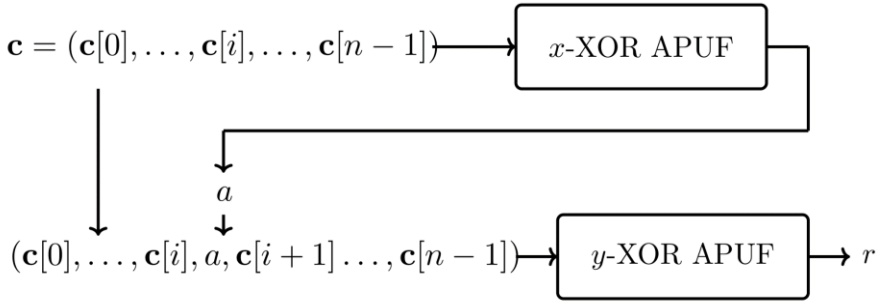


x-XOR PUF.

$$r_{\text{XOR APUF}} = sign(\prod_{i=1}^{x} w_i^{\mathrm{T}} \Phi) \quad [2]$$

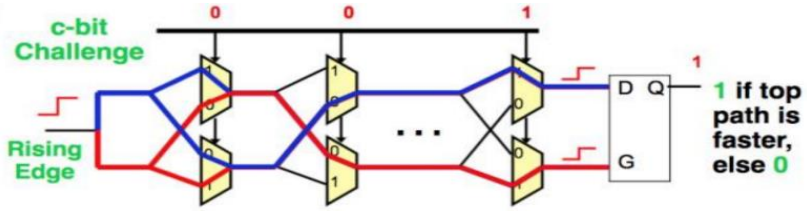
Precise non-linear model + CRPs + classical ML
= impractically softwarelly clonable

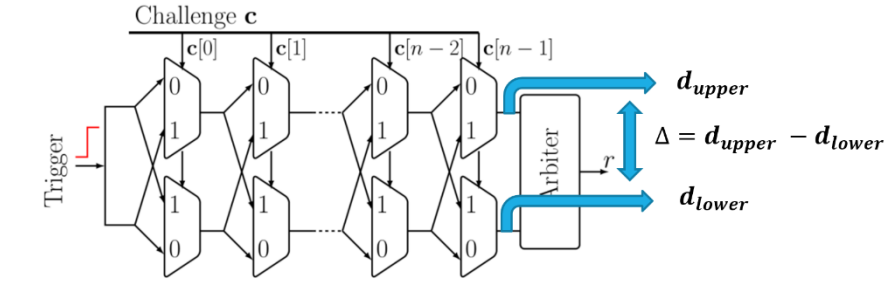
Interpose PUF (iPUF)



APUF, XOR APUF and iPUF

Arbiter PUF (APUF)





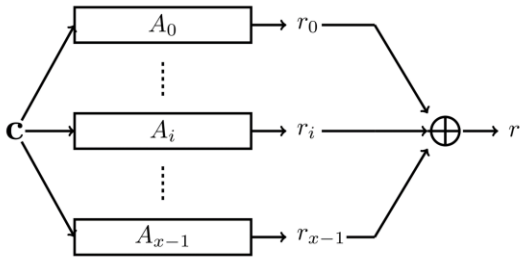
A n-stage classic Arbiter PUF with challenge $\mathbf{c} \in \{0,1\}^n$.

- $\Delta > 0 \rightarrow r = 1. Otherwise\ r = 0$
- $\Delta = \boldsymbol{d_{upper}} - \boldsymbol{d_{lower}} = \boldsymbol{w} \cdot \boldsymbol{\Phi}$
- $\boldsymbol{w}$: $unique\ for\ any\ APUF\ instance$
- $\boldsymbol{\Phi}\ is\ the\ parity\ vector$

$\boldsymbol{\Phi}[i] = \prod_{j=i,\dots,n-1}(1 - \boldsymbol{c}[j]), i = 0, \dots, n-1\ ,\ \boldsymbol{\Phi}[n] = 1$

Precise linear model + CRPs + ML
= practically and softwarelly clonable

x-XOR Arbiter PUF

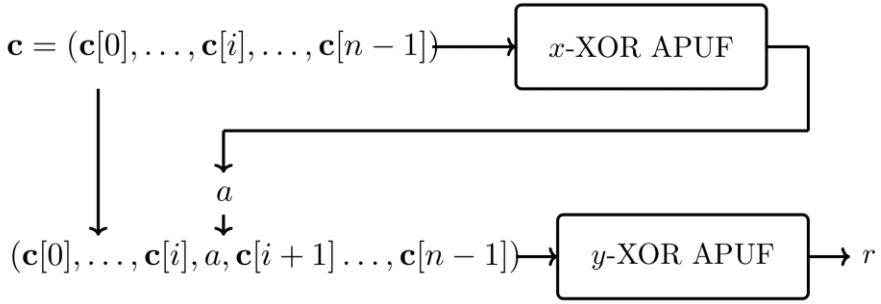


x-XOR PUF.

$$r_{\text{XOR APUF}} = sign(\prod_{i=1}^{x} w_i^{\mathrm{T}} \Phi)$$

Precise non-linear model + CRPs + classical ML
= impractically softwarelly clonable

Interpose PUF (iPUF)

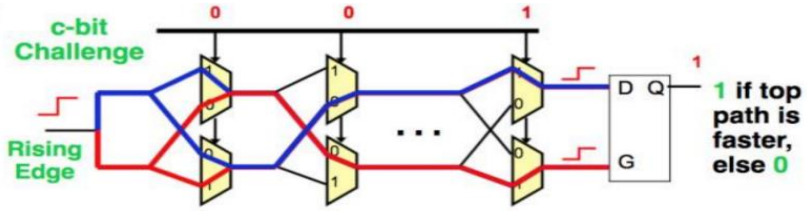


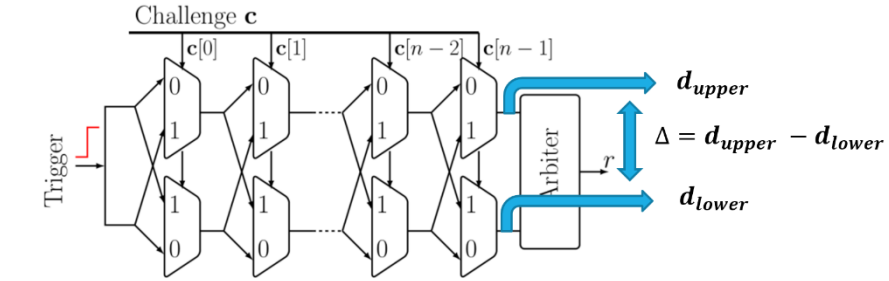
$$(x, y) - IPUF$$
$$\approx \left(y + \frac{x}{2}\right) - XOR\ PUF$$

if a is inserted at the middle

APUF, XOR APUF and iPUF

Arbiter PUF (APUF)





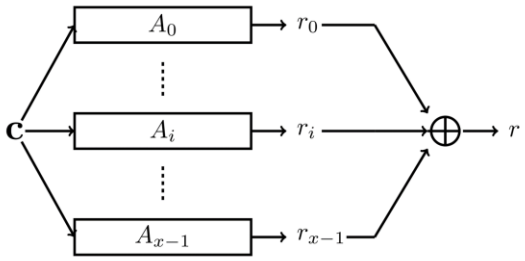
A n-stage classic Arbiter PUF with challenge $\mathbf{c} \in \{0,1\}^n$.

- $\Delta > 0 \rightarrow r = 1. Otherwise\ r = 0$
- $\Delta = \boldsymbol{d_{upper}} - \boldsymbol{d_{lower}} = \boldsymbol{w} \cdot \boldsymbol{\Phi}$
- $\boldsymbol{w}$: $unique\ for\ any\ APUF\ instance$
- $\boldsymbol{\Phi}$ $is\ the\ parity\ vector$

$\boldsymbol{\Phi}[i] = \prod_{j=i,\dots,n-1}(1 - \boldsymbol{c}[j]), i = 0, \dots, n-1\ ,\ \boldsymbol{\Phi}[n] = 1$

Precise linear model + CRPs + ML
= practically and softwarelly clonable

x-XOR Arbiter PUF

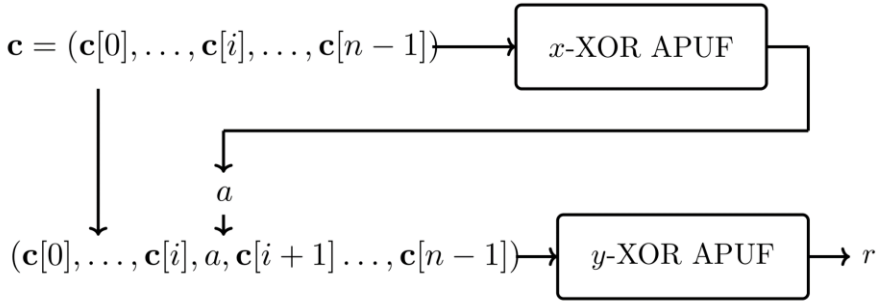


x-XOR PUF.

$$r_{\text{XOR APUF}} = sign(\prod_{i=1}^{x} w_i^{\mathrm{T}} \Phi)$$

Precise non-linear model + CRPs + classical ML
= impractically softwarelly clonable

Interpose PUF (iPUF)



$$(x, y) - IPUF \approx \left(y + \frac{x}{2}\right) - XOR\ PUF$$

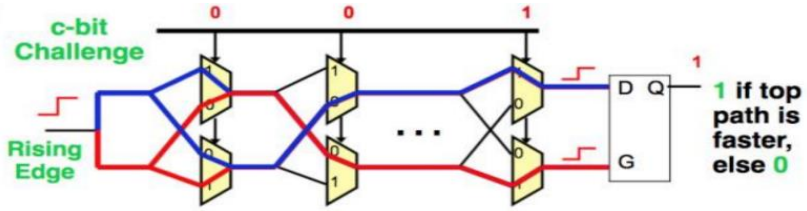
if a is inserted at the middle

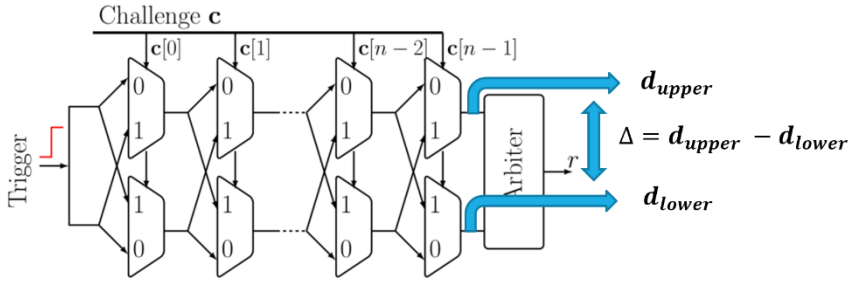
Precise non-linear model + CRPs + classical ML
= impractically softwarelly clonable

UCONN

APUF, XOR APUF and iPUF

Arbiter PUF (APUF)





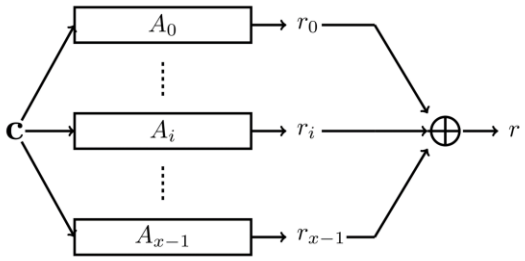
A n-stage classic Arbiter PUF with challenge $\mathbf{c} \in \{0,1\}^n$.

- $\Delta > 0 \rightarrow r = 1. Otherwise\ r = 0$
- $\Delta = \boldsymbol{d_{upper}} - \boldsymbol{d_{lower}} = \boldsymbol{w} \cdot \boldsymbol{\Phi}$
- $\boldsymbol{w}$: $unique\ for\ any\ APUF\ instance$
- $\boldsymbol{\Phi}$ $is\ the\ parity\ vector$

$\boldsymbol{\Phi}[i] = \prod_{j=i,\dots,n-1}(1 - \boldsymbol{c}[j]), i = 0, \dots, n-1\ ,\ \boldsymbol{\Phi}[n] = 1$

Precise linear model + CRPs + ML
= practically and softwarelly clonable

x-XOR Arbiter PUF

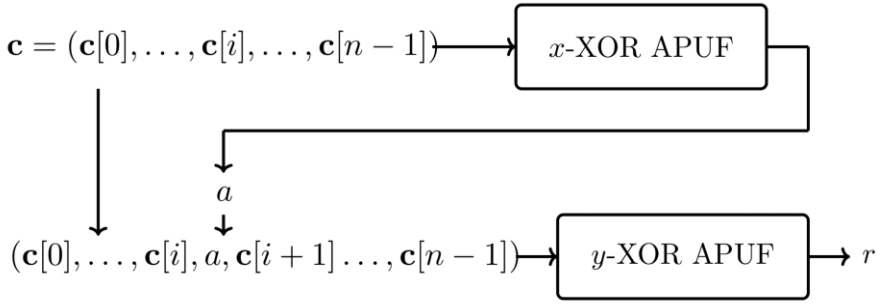


x-XOR PUF.

$$r_{\text{XOR APUF}} = sign(\prod_{i=1}^{x} w_i^{\mathrm{T}} \Phi)$$

Precise non-linear model + CRPs + classical ML
= impractically softwarelly clonable

Interpose PUF (iPUF)



$$(x, y) - IPUF \approx \left(y + \frac{x}{2}\right) - XOR\ PUF$$

if a is inserted at the middle

Precise non-linear model + CRPs + classical ML
= impractically softwarelly clonable

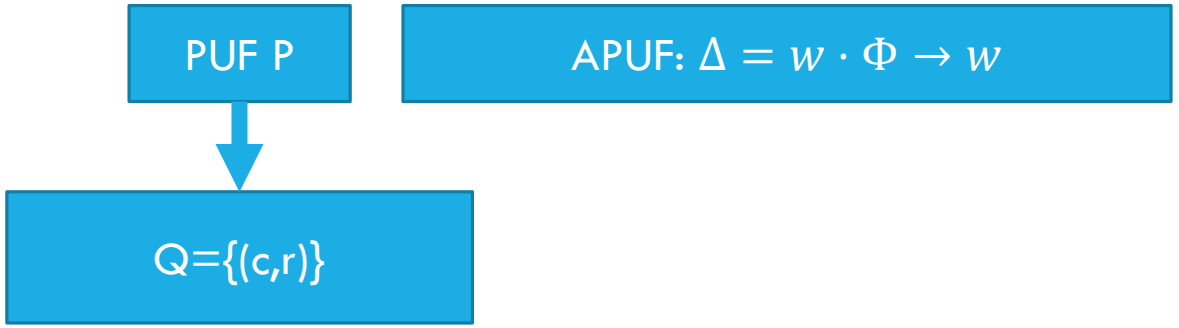
XOR APUF is not Secure against noisy CRPs + CMA-ES [Advanced ML]! (CHES2015) why? Why not for IPUF?

UCONN

3. Machine Learning based Modeling Attacks on PUFs

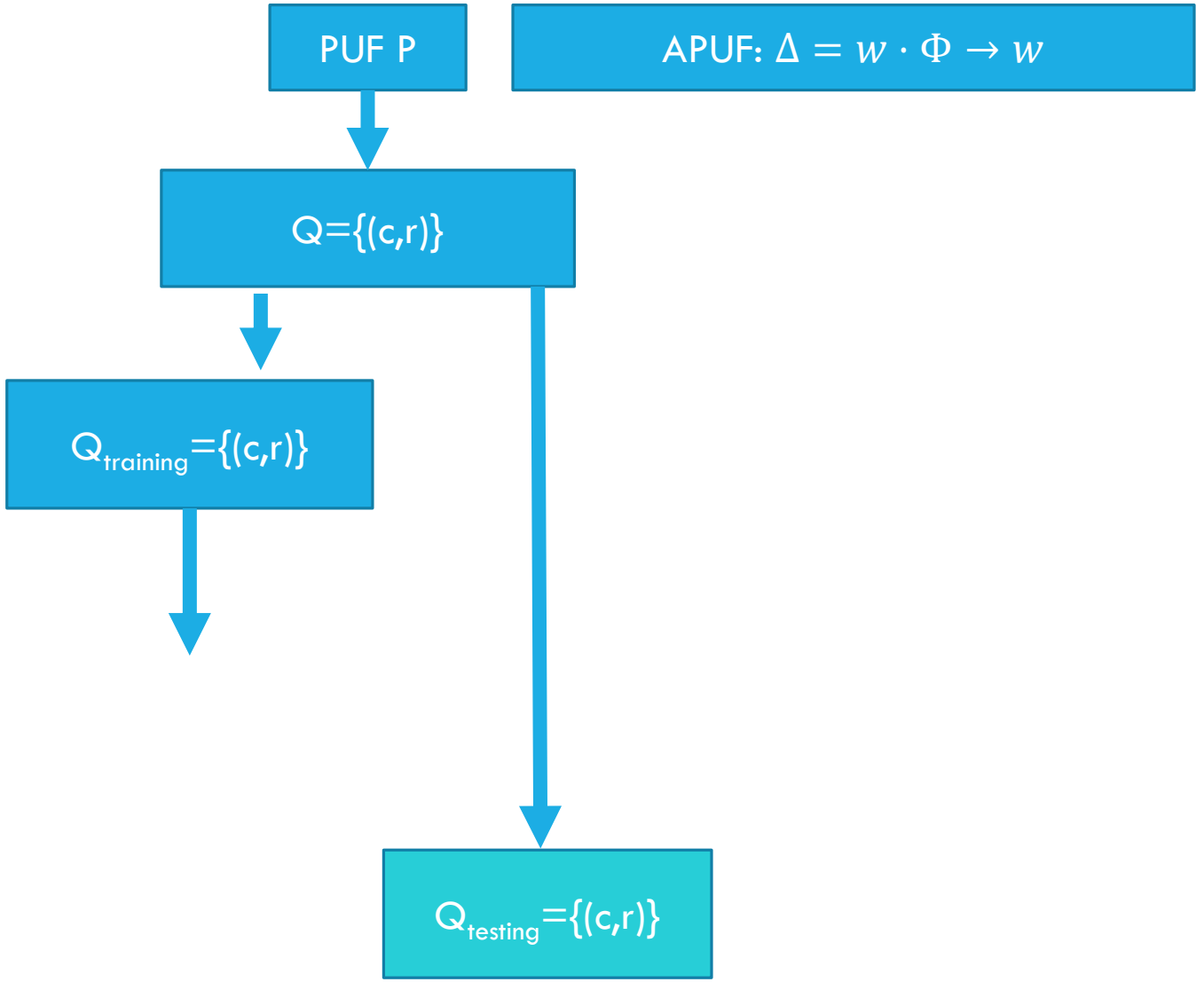
UCONN

Basic Steps in MLMA



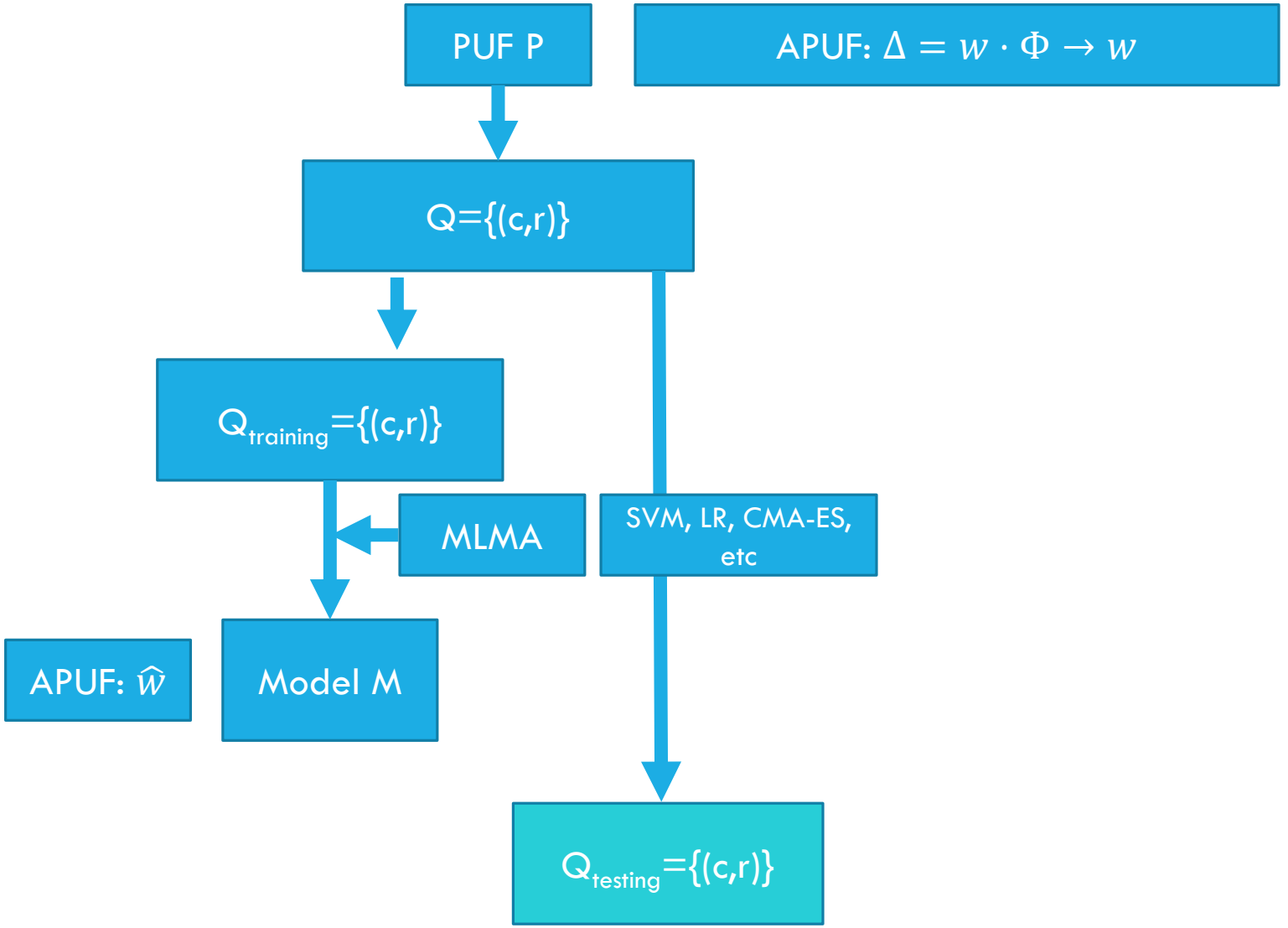
UCONN

Basic Steps in MLMA



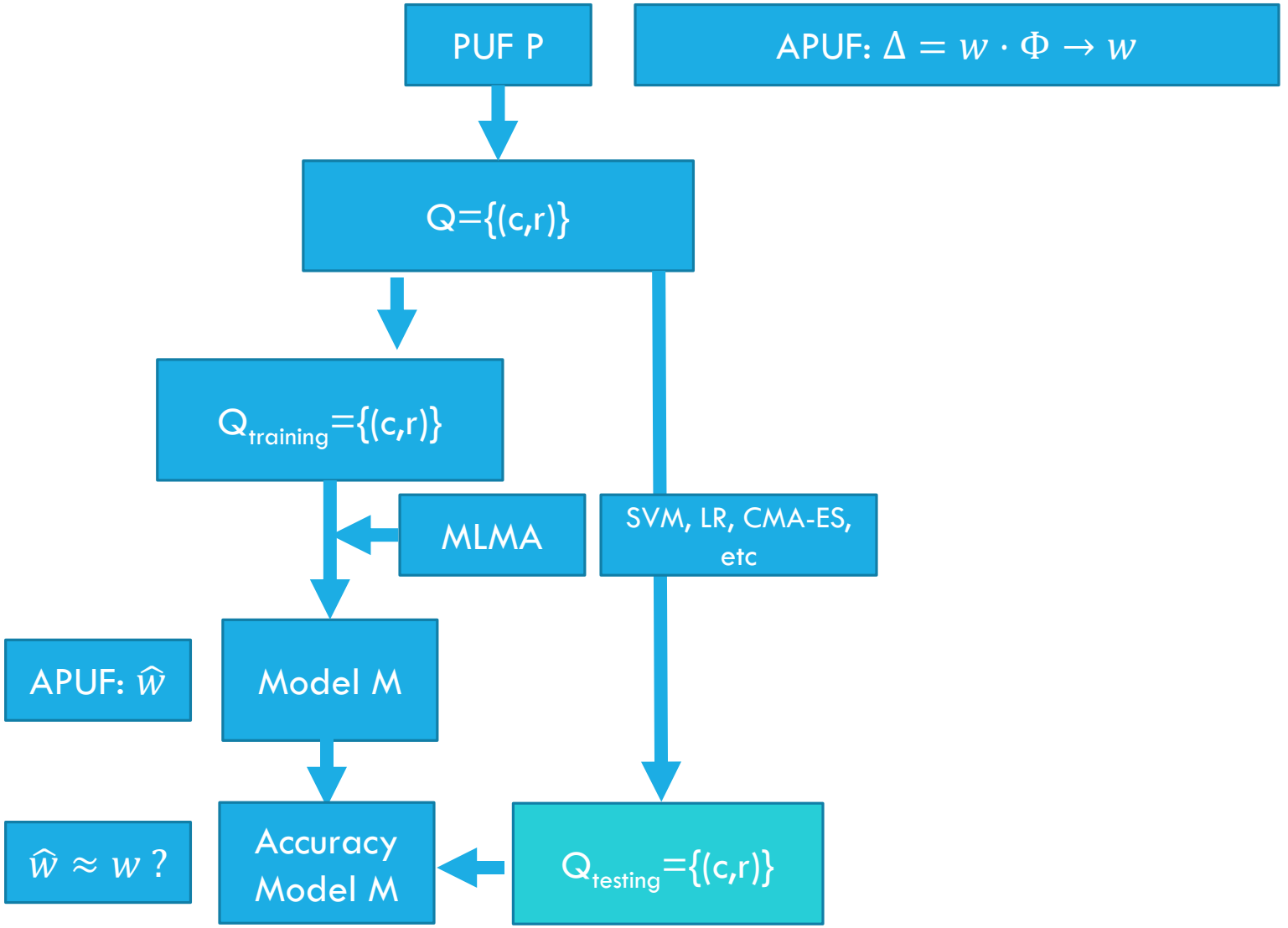
UCONN

Basic Steps in MLMA



UCONN

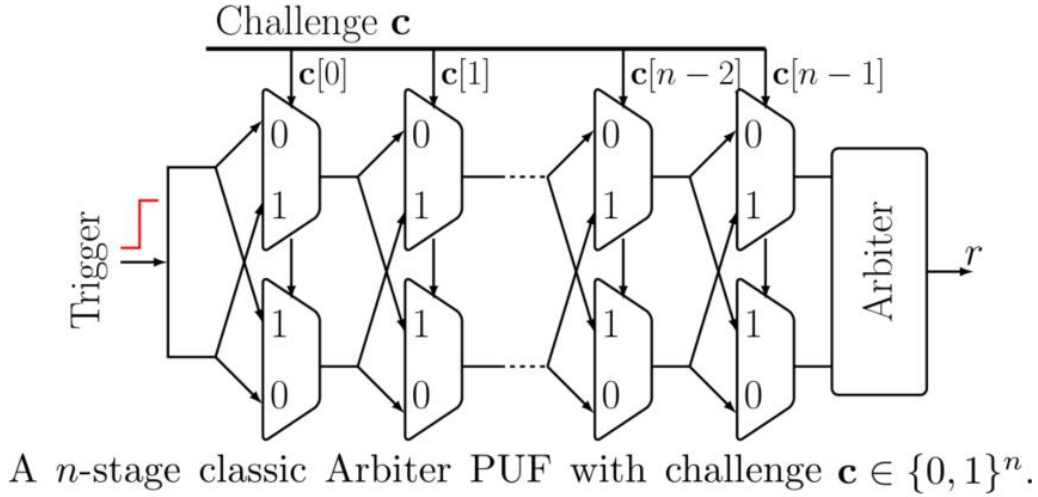
Basic Steps in MLMA



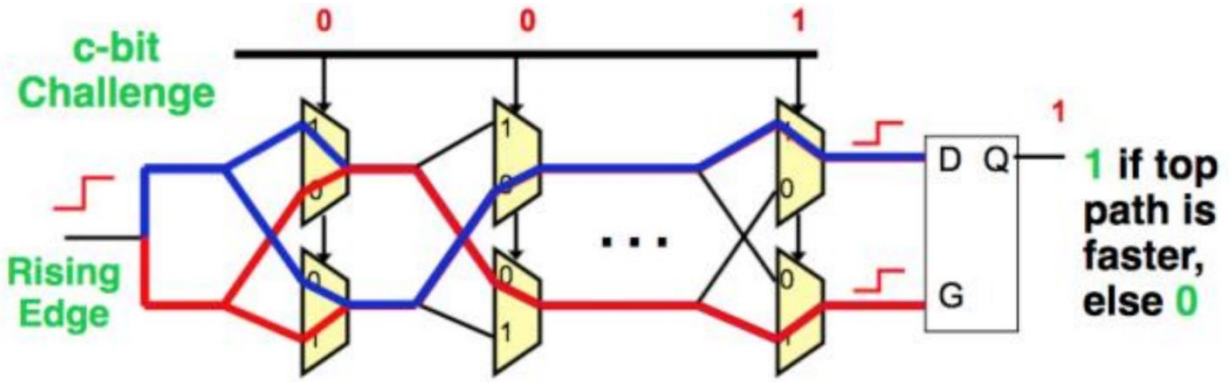
UCONN

4. Classical Machine Learning Attacks on APUFs

Arbiter PUF



A n-stage classic Arbiter PUF with challenge $\mathbf{c} \in \{0,1\}^n$.

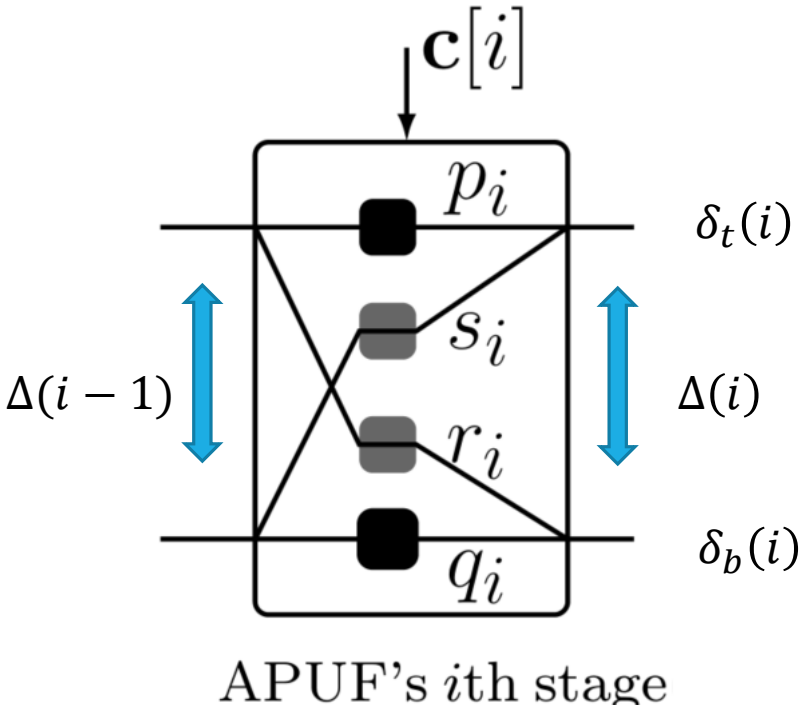


[10]

- Blaise Gassend, Dwaine E. Clarke, Marten van Dijk, Srinivas Devadas: Silicon physical random functions. ACM Conference on Computer and Communications Security 2002: 148-160

UCONN

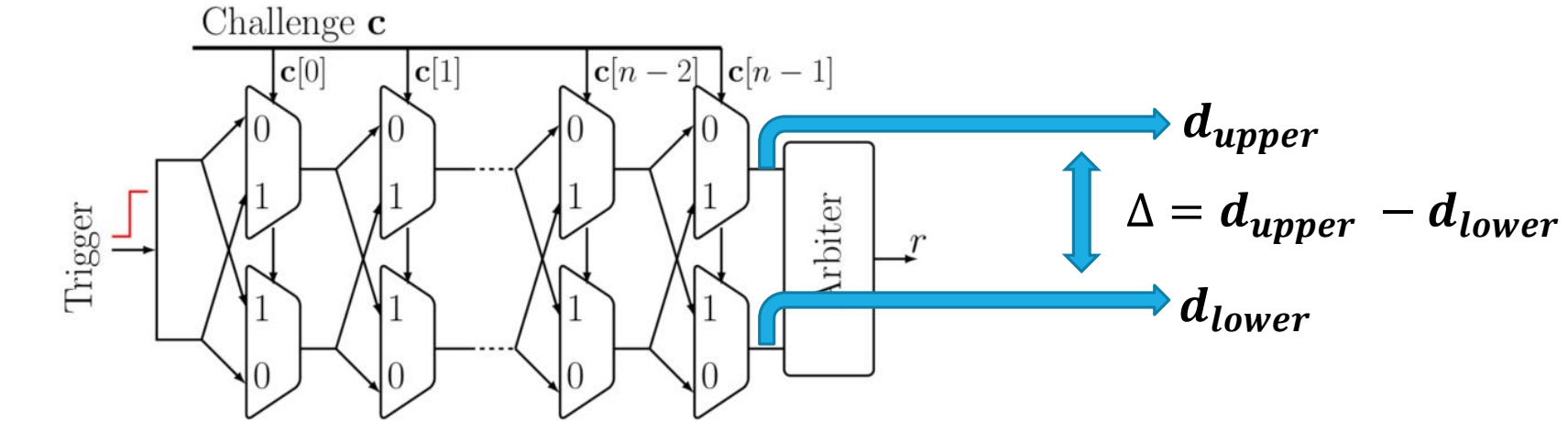
Delays in APUFs



APUF's ith stage

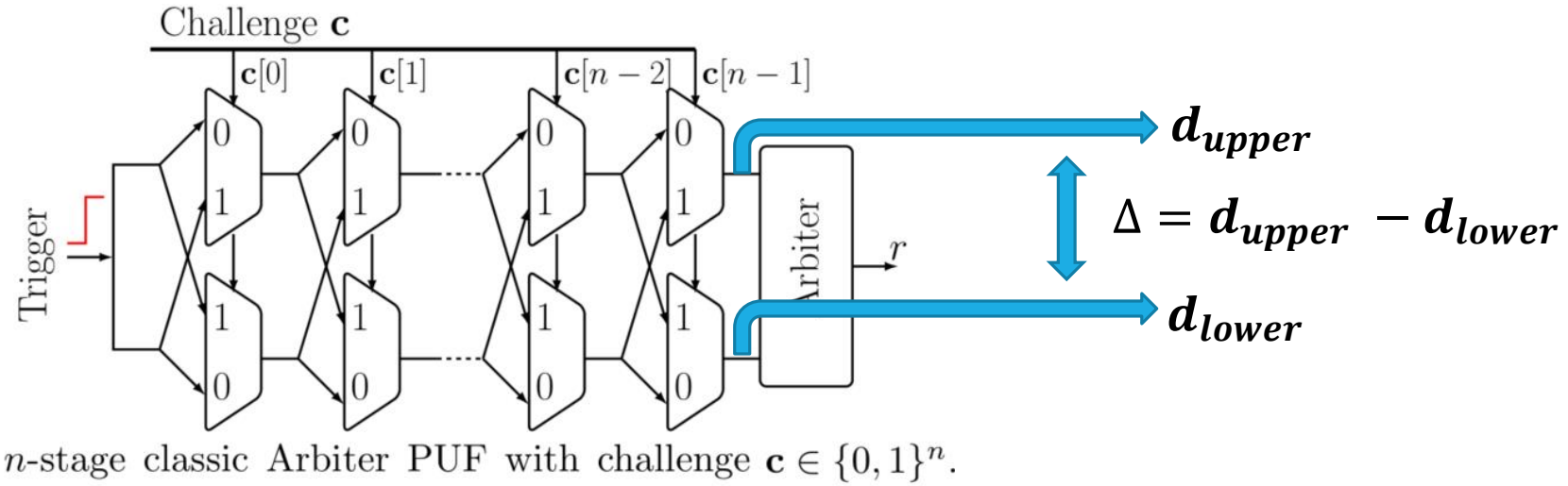
$$\Delta(i) = \delta_t(i) - \delta_b(i)$$
$$= \Delta(i-1)\mathbf{c}[i] + \alpha_i \mathbf{c}[i] + \beta_i$$
$$\alpha_i = \frac{p_i - q_i + r_i - s_i}{2}$$
$$\beta_i = \frac{p_i - q_i - r_i + s_i}{2}$$
$$\mathbf{c}[i] \in \{0, 1\}$$

APUF linear delay model



A n-stage classic Arbiter PUF with challenge $\mathbf{c} \in \{0,1\}^n$.

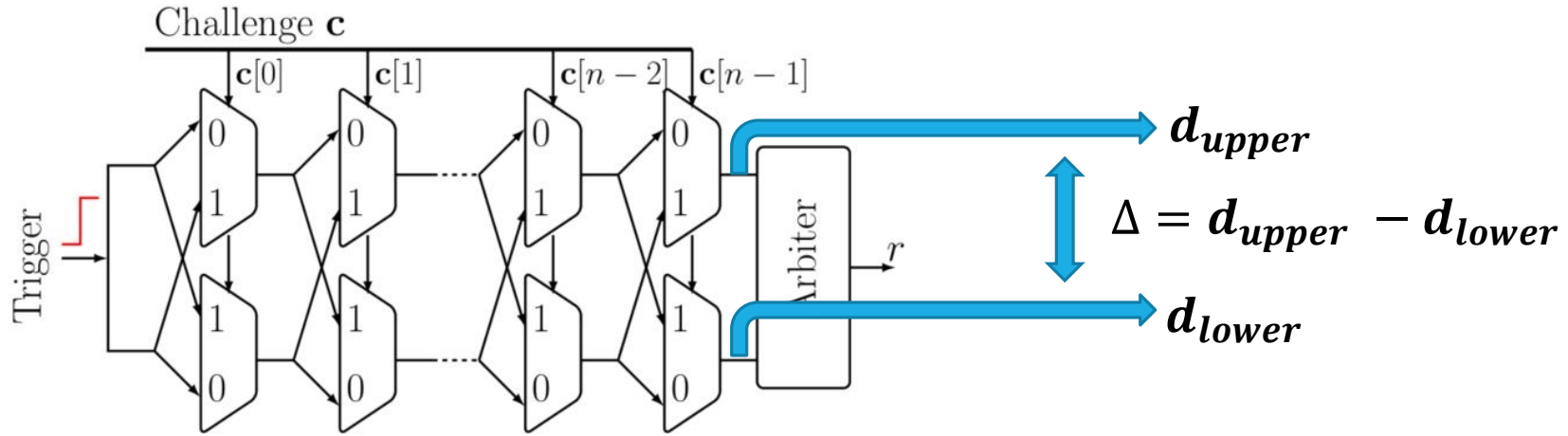
APUF linear delay model



A n-stage classic Arbiter PUF with challenge $\mathbf{c} \in \{0,1\}^n$.

- $\Delta > 0 \rightarrow r = 1. Otherwise\ r = 0$

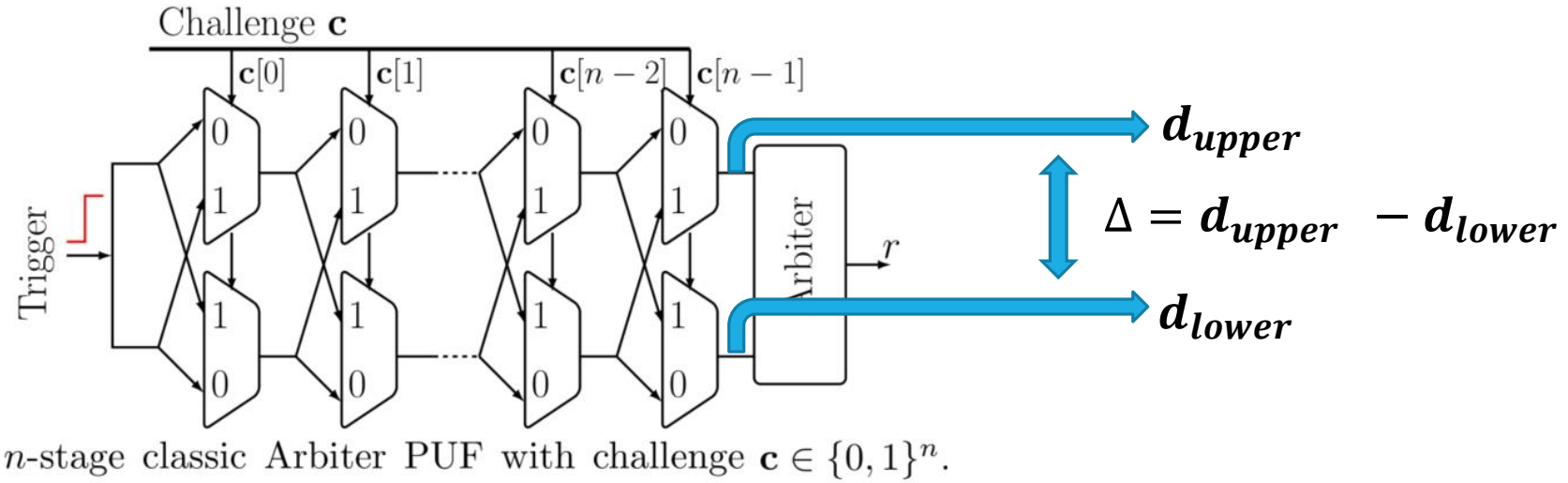
APUF linear delay model



A n-stage classic Arbiter PUF with challenge $\mathbf{c} \in \{0,1\}^n$.

- $\Delta > 0 \rightarrow r = 1. Otherwise\ r = 0$
- $\Delta = \boldsymbol{d_{upper}} - \boldsymbol{d_{lower}} = \boldsymbol{w} \cdot \boldsymbol{\Phi}$
- $\boldsymbol{w}$ *is the weight vector, unique for each APUF instance and represents the delay of a given APUF instance*

APUF linear delay model



A n-stage classic Arbiter PUF with challenge $\mathbf{c} \in \{0,1\}^n$.

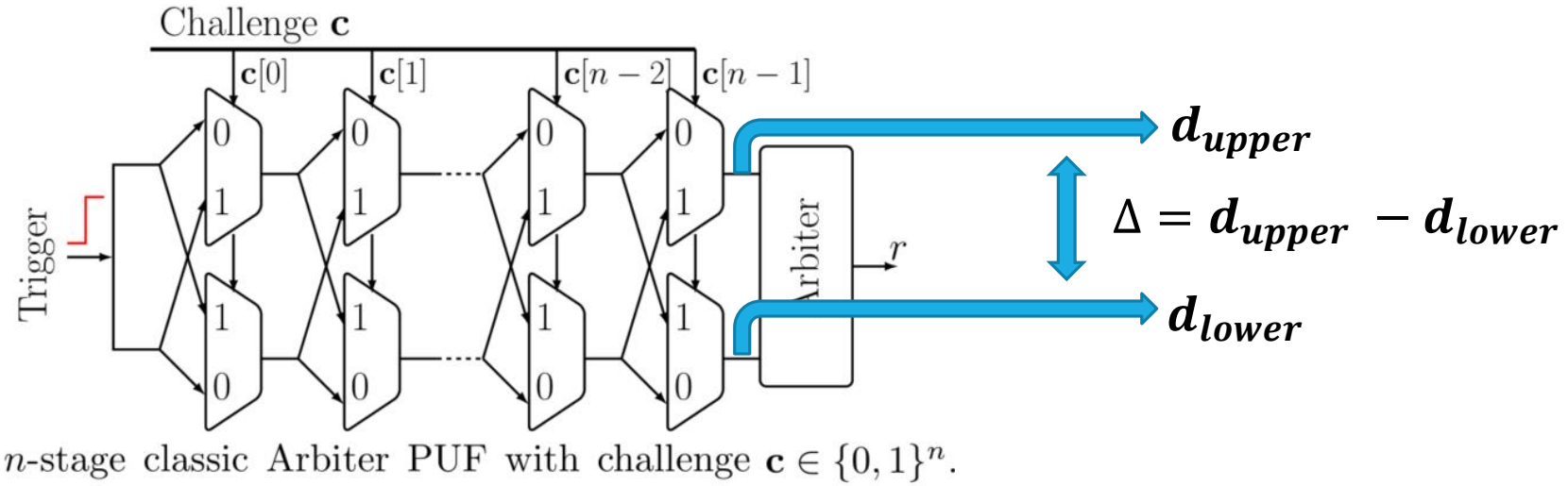
- $\Delta > 0 \rightarrow r = 1. Otherwise\ r = 0$
- $\Delta = \boldsymbol{d_{upper}} - \boldsymbol{d_{lower}} = \boldsymbol{w} \cdot \boldsymbol{\Phi}$
- $\boldsymbol{w},\ \boldsymbol{w}[\boldsymbol{i}] = \alpha_{i+1} + \beta_i, i = 1, \cdots, n-1,\ \boldsymbol{w}[\boldsymbol{0}] = \alpha_0, \boldsymbol{w}[\boldsymbol{n}] = \beta_n$
- $\boldsymbol{\Phi}$ *is the parity vector*

$$\boldsymbol{\Phi}[i] = \prod_{j=i,\ldots,n-1} (1 - \boldsymbol{c}[j]), i = 0, \ldots, n-1$$

$$\boldsymbol{\Phi}[n] = 1$$

UCONN

APUF linear delay model



A n-stage classic Arbiter PUF with challenge $\mathbf{c} \in \{0,1\}^n$.

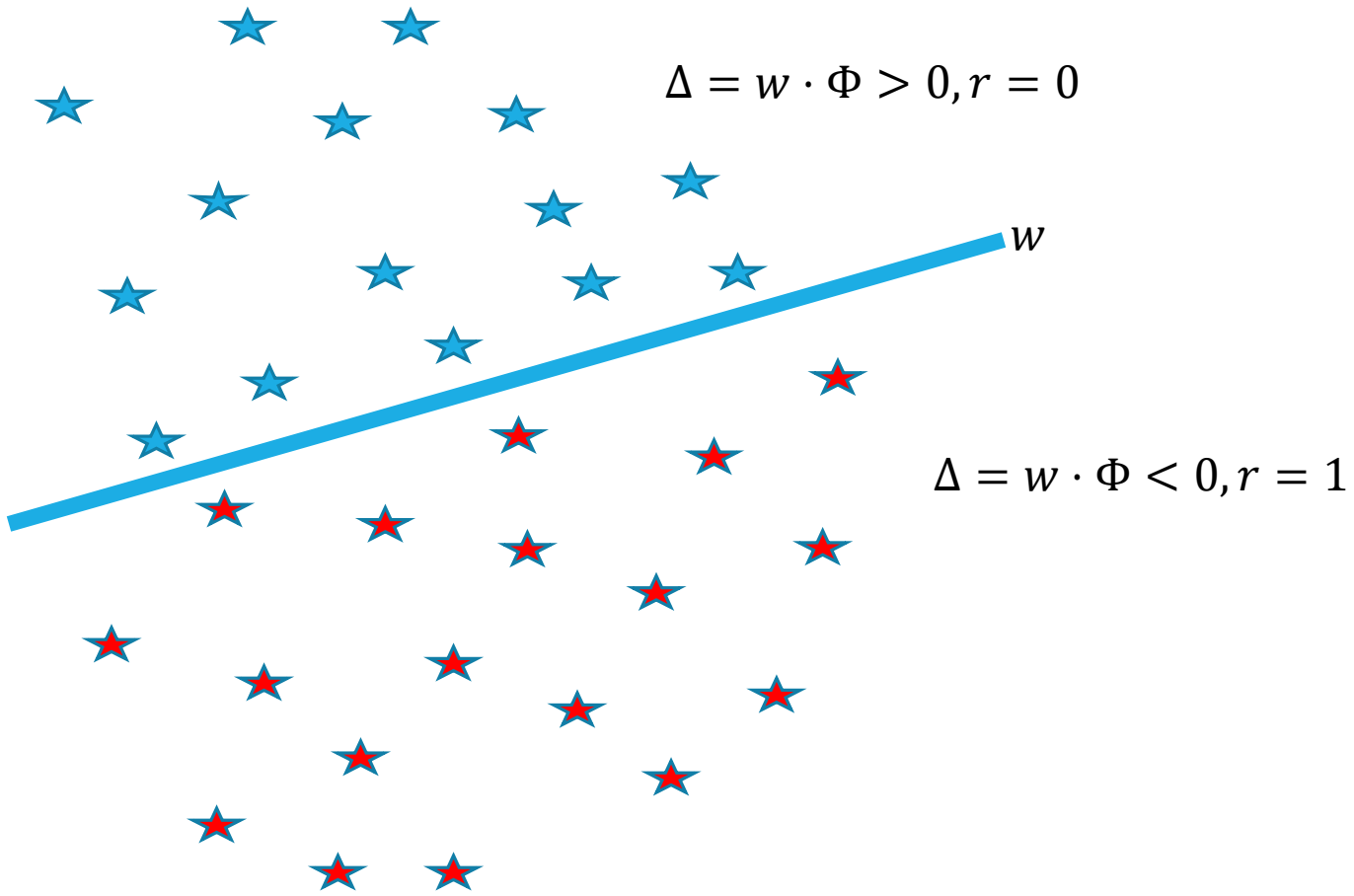
- $\Delta > 0 \rightarrow r = 1. Otherwise\ r = 0$
- $\Delta = \boldsymbol{d_{upper}} - \boldsymbol{d_{lower}} = \boldsymbol{w} \cdot \boldsymbol{\Phi}$
- $\boldsymbol{w}$ *is the weight vector, unique for each APUF instance and represents the delay of a given APUF instance*
- $\boldsymbol{\Phi}$ *is the parity vector*

$$\boldsymbol{\Phi}[i] = \prod_{j=i,\dots,n-1} (1 - \boldsymbol{c}[j]), i = 0, \dots, n-1$$

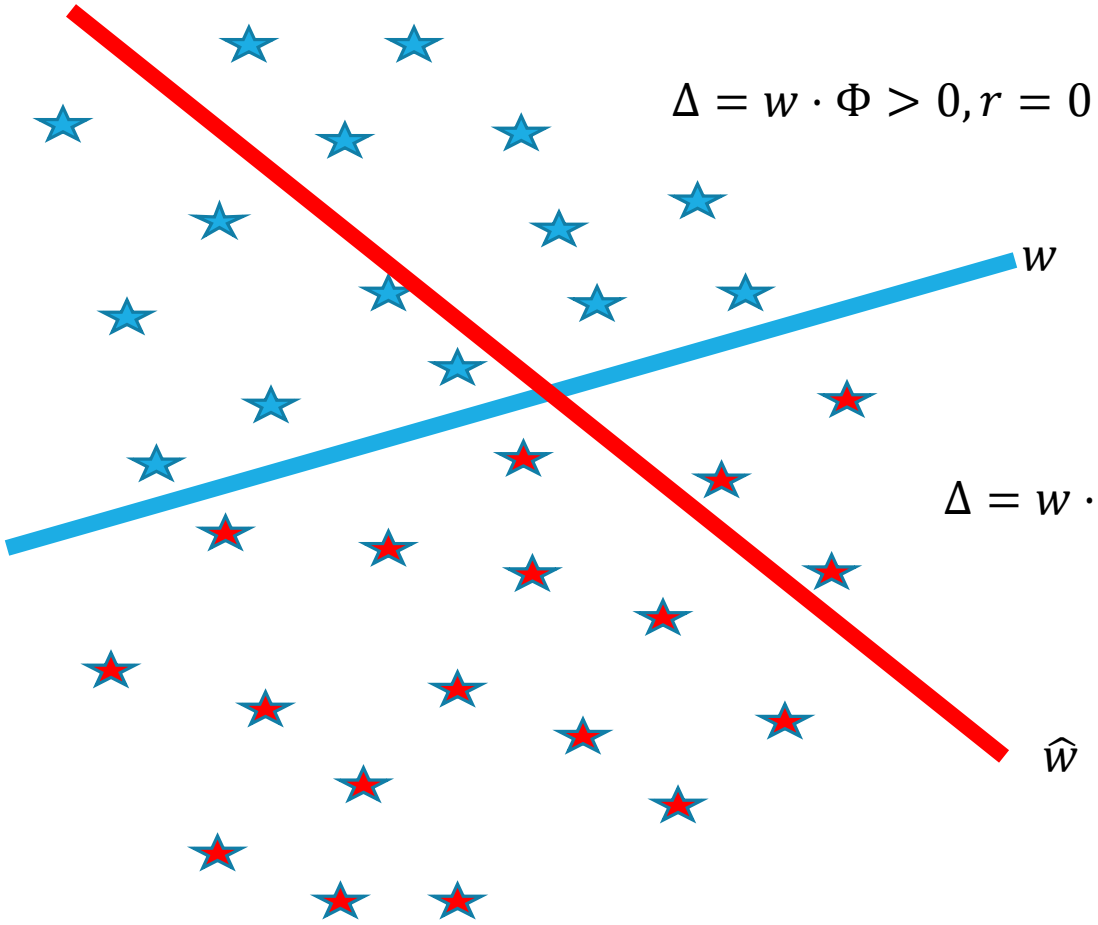
$$\boldsymbol{\Phi}[n] = 1$$

Security: it can be mathematically modeled if the adversary can access a set of CRPs of its by using machine learning techniques

Machine Learning based modeling attack on APUF



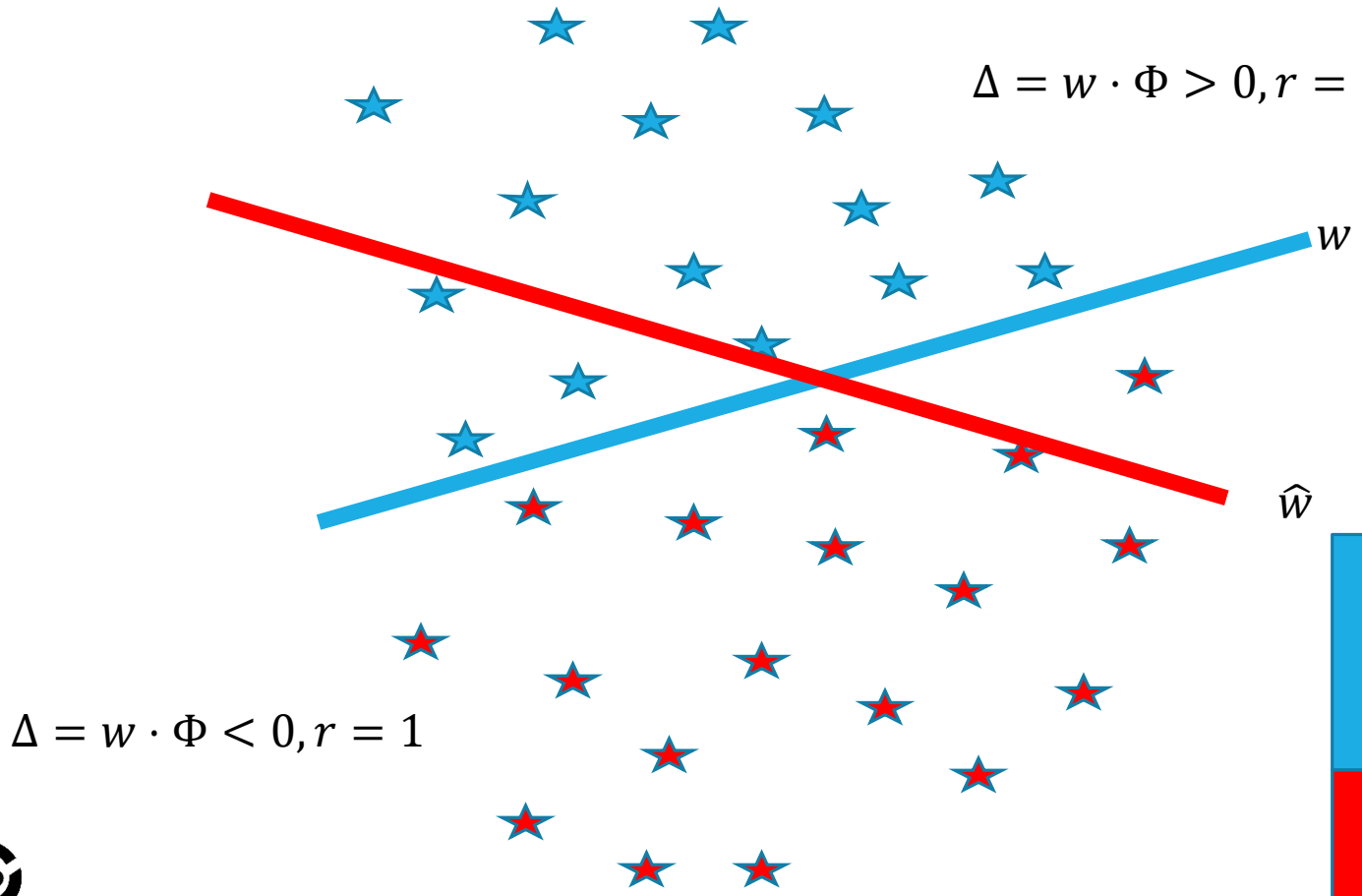
Machine Learning based modeling attack on APUF



Iteration 1

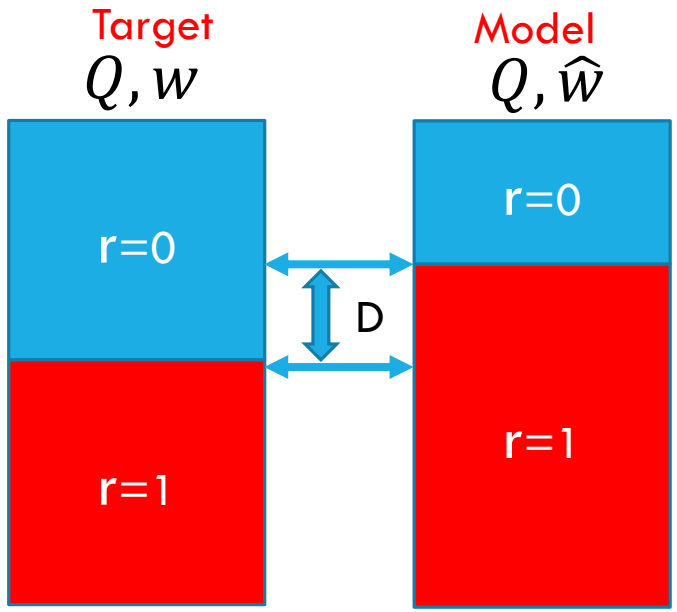
$\widehat{w}$ is randomly generated

Machine Learning based modeling attack on APUF



Iteration 2

$\widehat{w}$ is modified by some optimization method to reduce the gap D

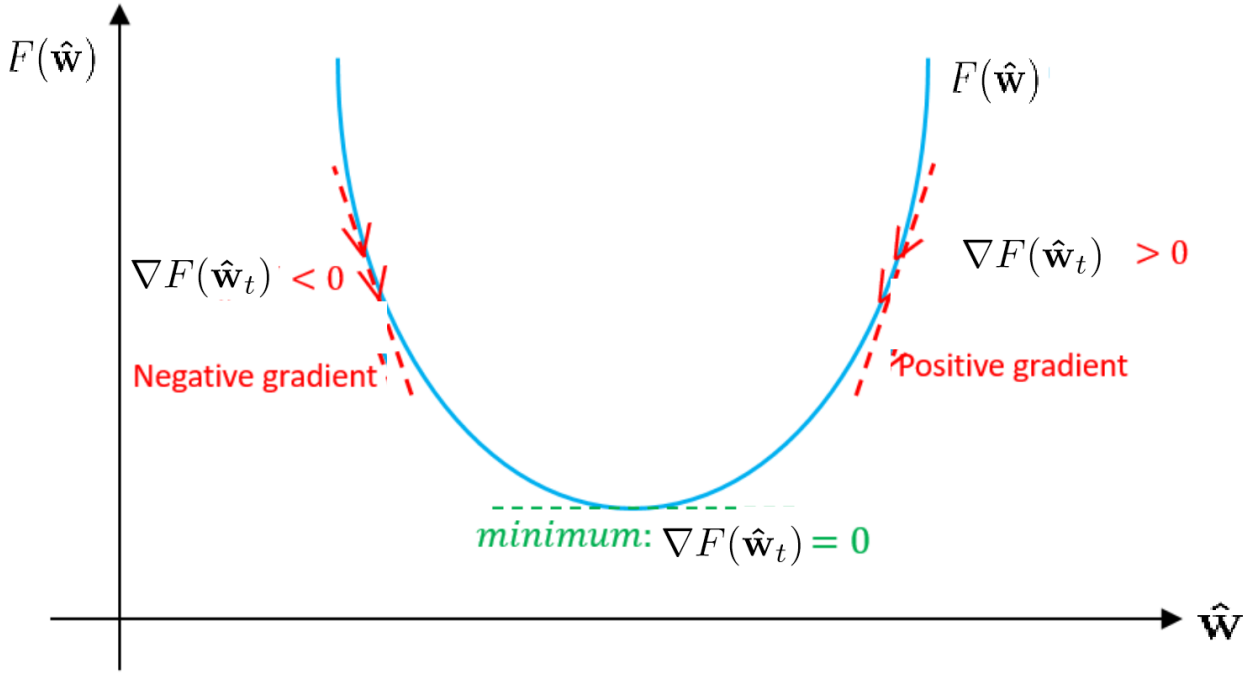


Machine Learning based modeling attack on APUF

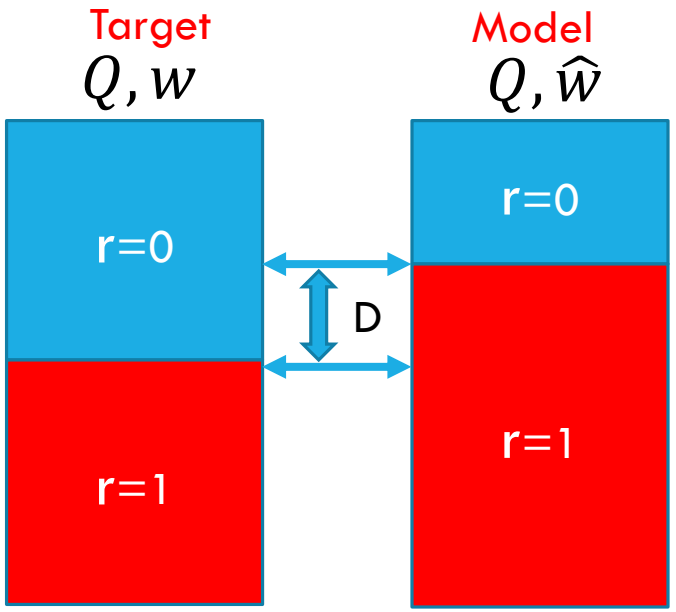
$$\min_{\hat{\mathbf{w}}} F(\hat{\mathbf{w}}) = \frac{1}{N}\sum_{i=1}^{N} L(\hat{\mathbf{w}}, (\mathbf{c}_i, r_i)),$$

where L is the loss function and $\{(\mathbf{c}_i, r_i)_{i=1,\ldots,N}\}$ is the set of N CRPs.

To find $\hat{\mathbf{w}}$, gradient based optimization methodology can be applied, e.g.,



$\hat{w}$ is modified by some optimization method to reduce the gap D



UCONN

Machine Learning based modeling attack on APUF

$$\min_{\hat{\mathbf{w}}} F(\hat{\mathbf{w}}) = \frac{1}{N}\sum_{i=1}^{N} L(\hat{\mathbf{w}}, (\mathbf{c}_i, r_i)),$$

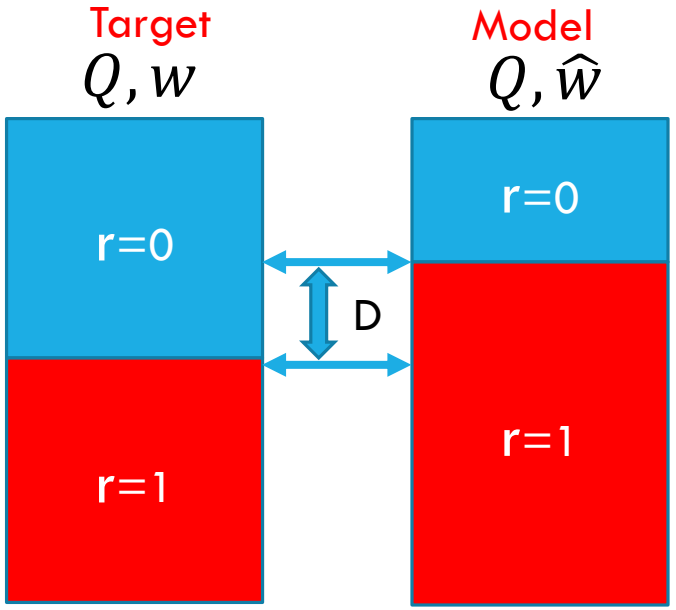
where L is the loss function and $\{(\mathbf{c}_i, r_i)_{i=1,\ldots,N}\}$ is the set of N CRPs.

To find $\hat{\mathbf{w}}$, gradient based optimization methodology can be applied, e.g.,

$$\hat{\mathbf{w}}_{t+1} = \hat{\mathbf{w}}_t - \eta \nabla F(\hat{\mathbf{w}}_t)$$

$$\nabla F(\hat{\mathbf{w}}_t) = \left(\frac{\partial F}{\partial \hat{\mathbf{w}}_t[0]}, \frac{\partial F}{\partial \hat{\mathbf{w}}_t[1]}, \ldots, \frac{\partial F}{\partial \hat{\mathbf{w}}_t[n-1]}\right)$$

$\hat{w}$ is modified by some optimization method to reduce the gap D



UCONN

Machine Learning based modeling attack on APUF

$$\min_{\hat{\mathbf{w}}} F(\hat{\mathbf{w}}) = \frac{1}{N}\sum_{i=1}^{N} L(\hat{\mathbf{w}}, (\mathbf{c}_i, r_i)),$$

where L is the loss function and $\{(\mathbf{c}_i, r_i)_{i=1,\ldots,N}\}$ is the set of N CRPs.

To find $\hat{\mathbf{w}}$, gradient based optimization methodology can be applied, e.g.,

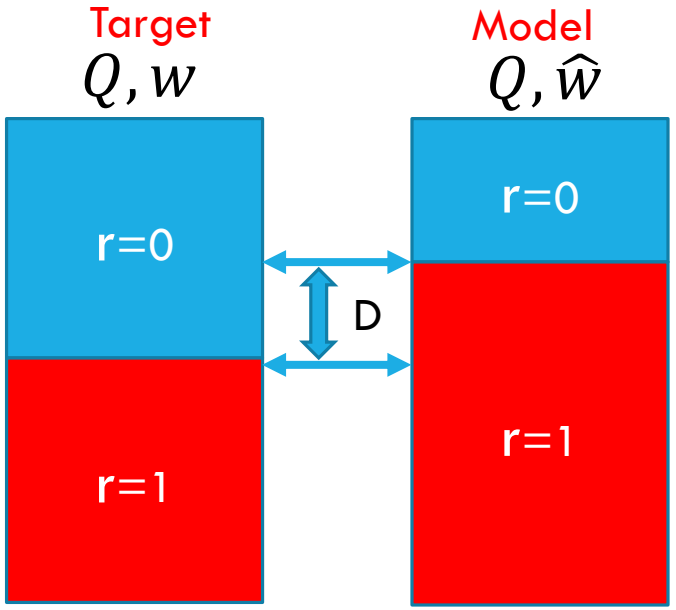
$$\hat{\mathbf{w}}_{t+1} = \hat{\mathbf{w}}_t - \eta \nabla F(\hat{\mathbf{w}}_t)$$

$$\nabla F(\hat{\mathbf{w}}_t) = \left(\frac{\partial F}{\partial \hat{\mathbf{w}}_t[0]}, \frac{\partial F}{\partial \hat{\mathbf{w}}_t[1]}, \ldots, \frac{\partial F}{\partial \hat{\mathbf{w}}_t[n-1]}\right)$$

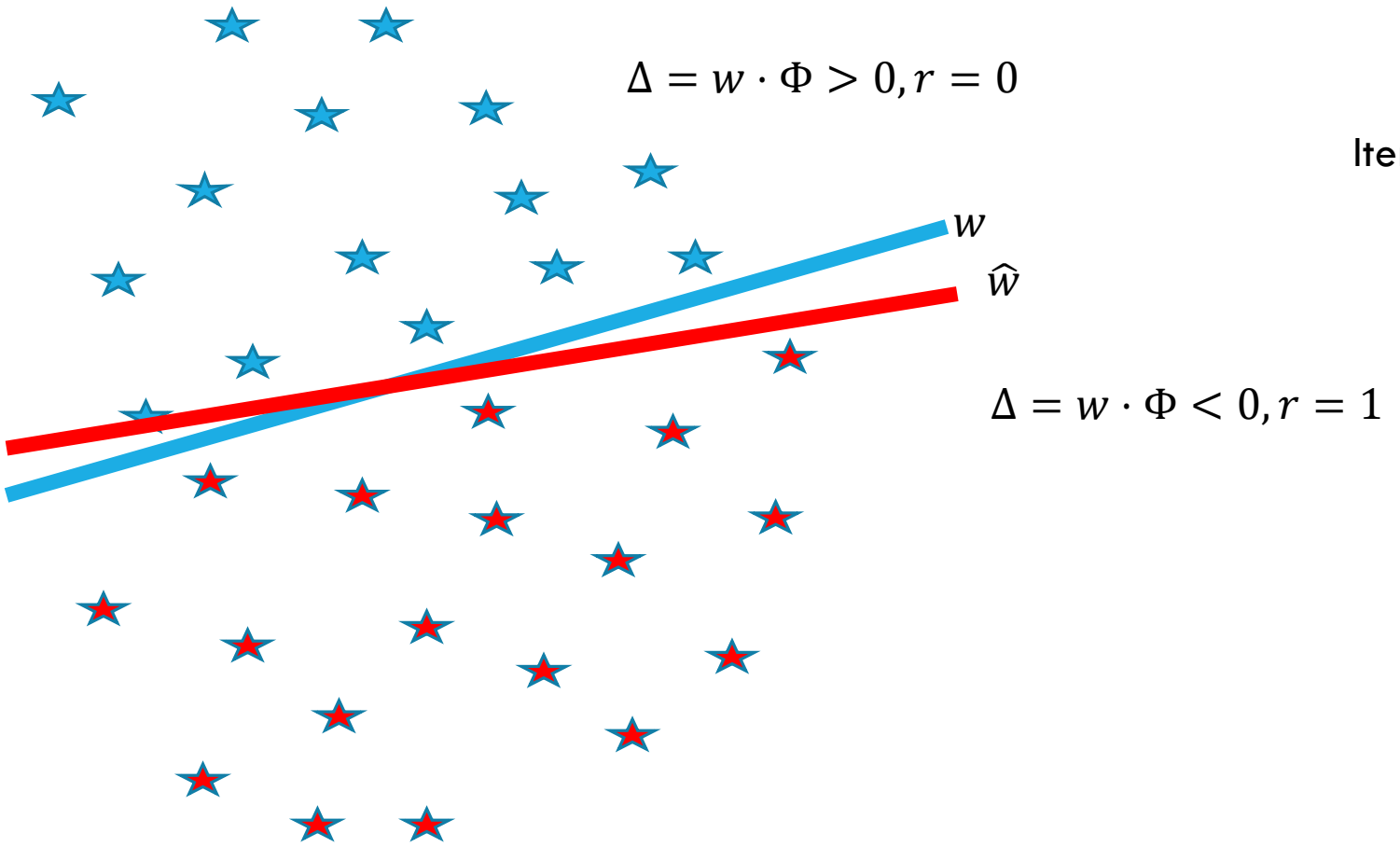
For example, $L(\hat{\mathbf{w}}, (\mathbf{c}_i, r_i)) = \frac{1}{2}\|\hat{\mathbf{w}}\mathbf{\Phi}(\mathbf{c_i}) - r_i\|^2$ then

$$\frac{\partial F}{\partial \hat{\mathbf{w}}_t[k]} = \sum_{i=1}^{N} (\hat{\mathbf{w}}[k]\mathbf{\Phi}(\mathbf{c_i}))[k] - r_i) \times \mathbf{\Phi}(\mathbf{c_i}))[k],\ k = 0, \ldots, n-1.$$

$\hat{w}$ is modified by some optimization method to reduce the gap D

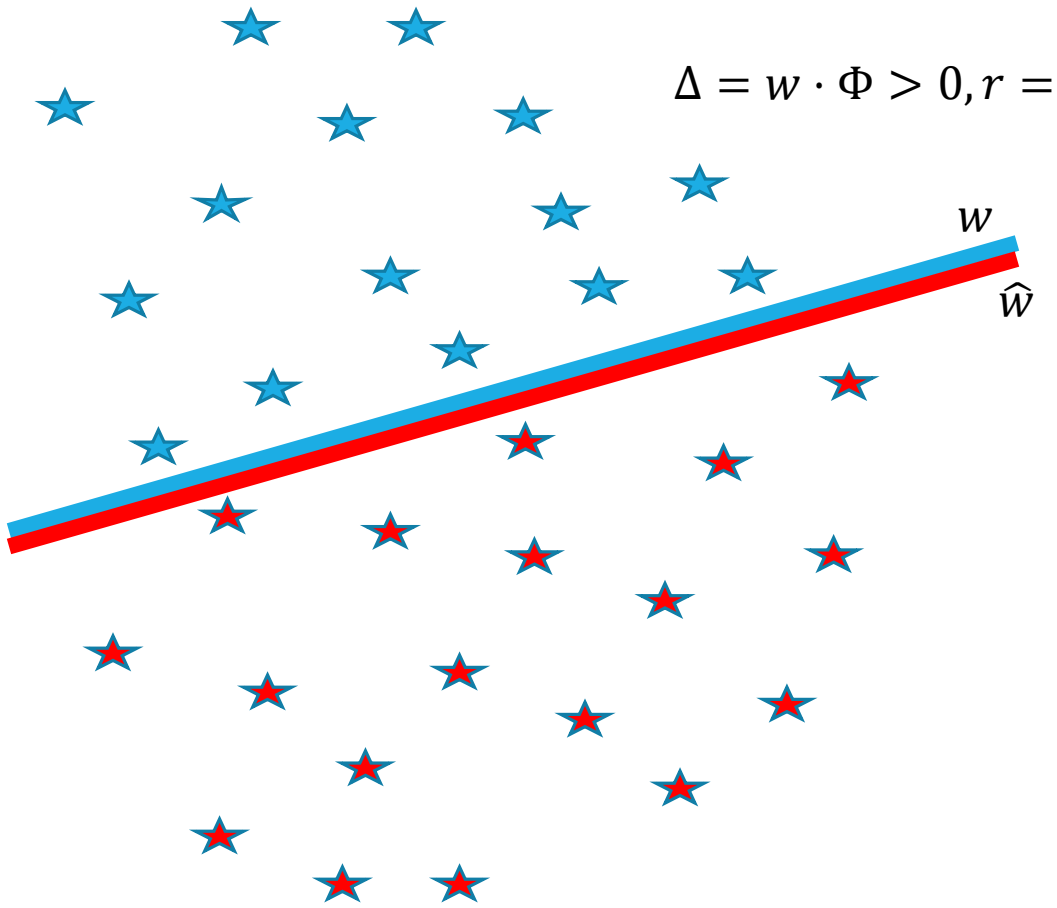


Machine Learning based modeling attack on APUF



Iteration n-1

Machine Learning based modeling attack on APUF



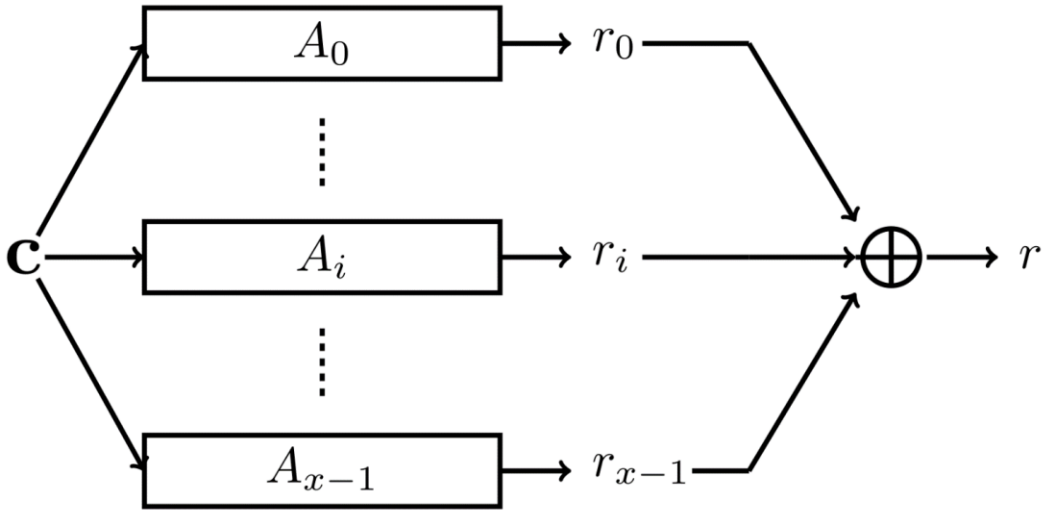
Iteration n

UCONN

5. Classical Machine Learning Attacks on XOR APUFs

UCONN

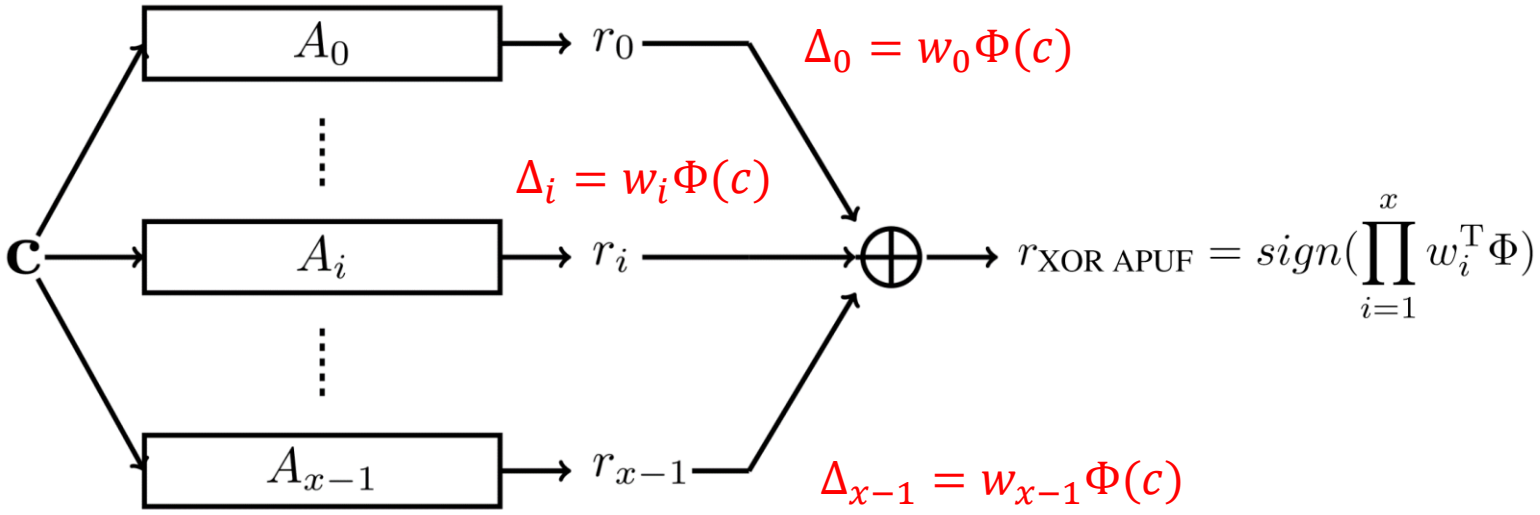
x-XOR APUF



x-XOR PUF.

Physical unclonable functions for device authentication and secret key generation. DAC2007, G. E. Suh and S. Devadas

x-XOR APUF

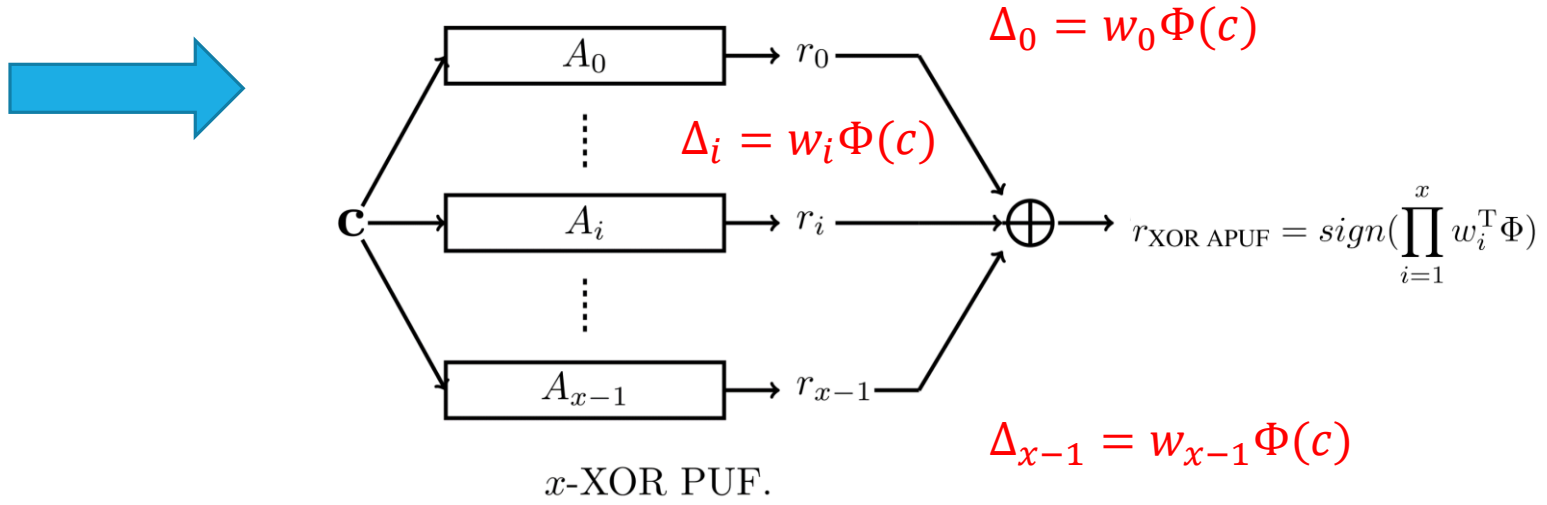


x-XOR PUF.

Physical unclonable functions for device authentication and secret key generation. DAC2007, G. E. Suh and S. Devadas

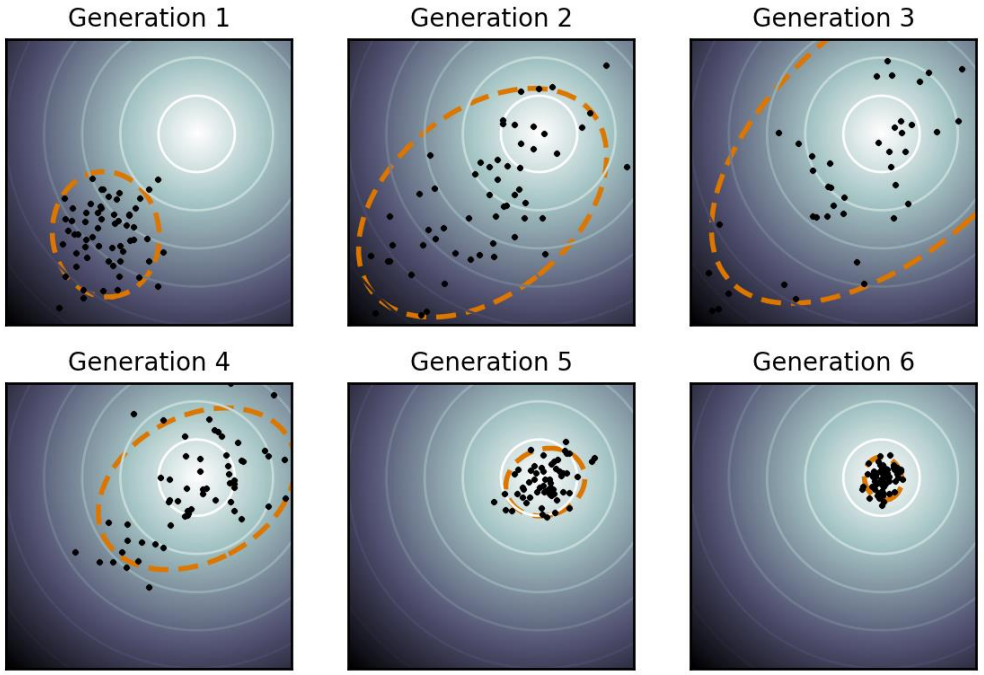
Attacks on x-XOR APUF: Evolution Strategy

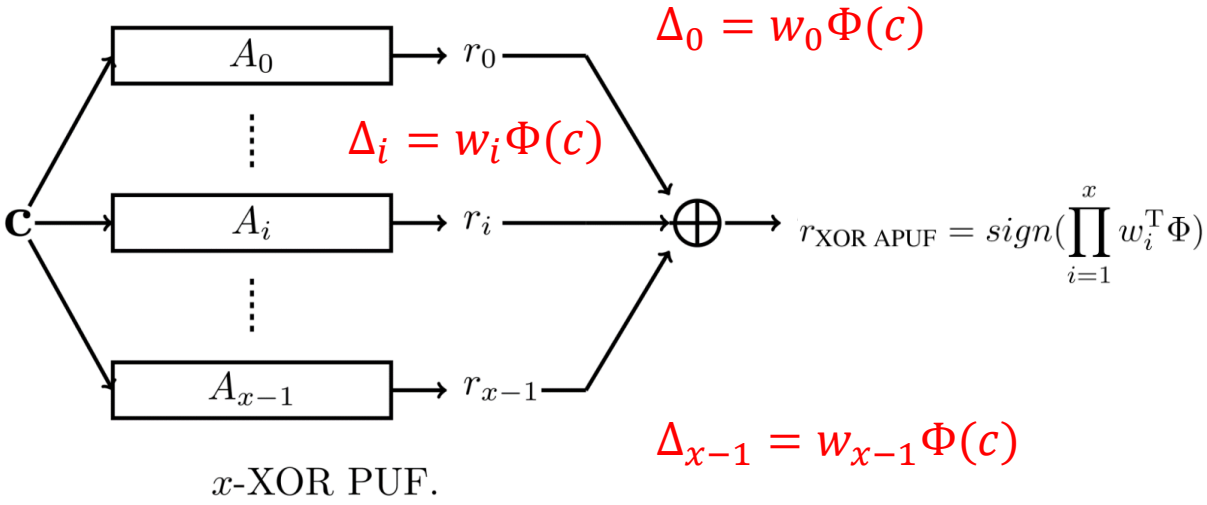
Evolution Strategy based ML



Attacks on x-XOR APUF: Evolution Strategy

Evolution Strategy based ML

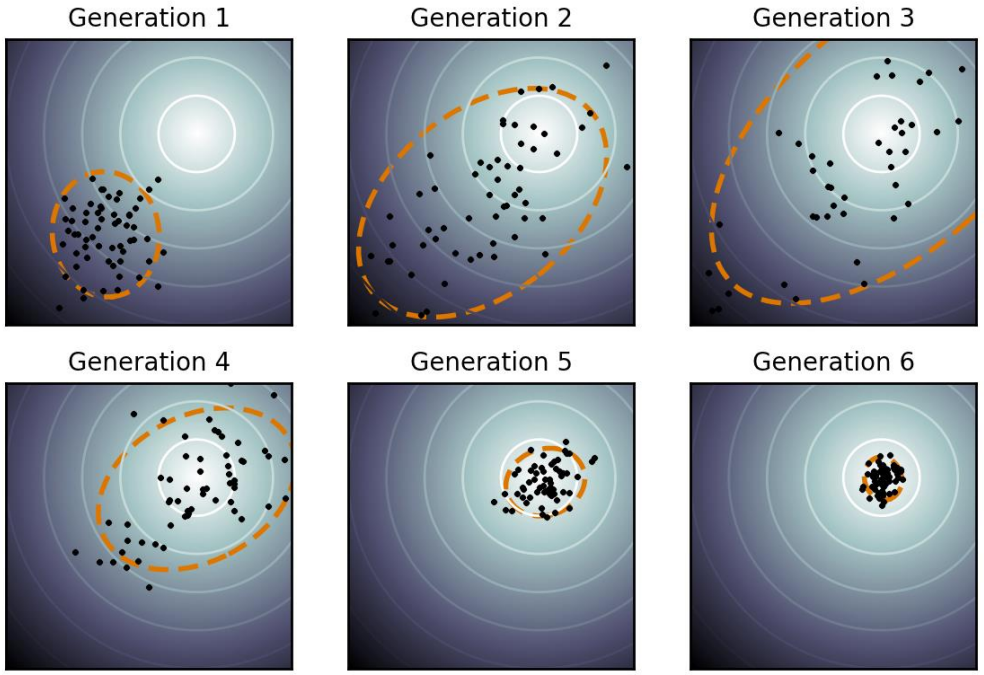


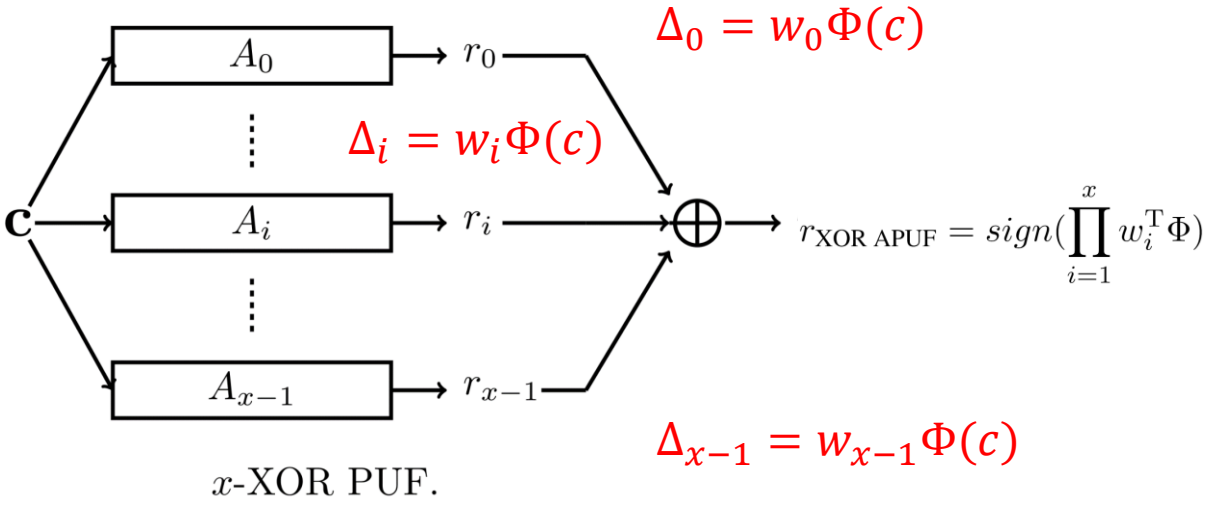


x-XOR PUF.

Attacks on x-XOR APUF: Evolution Strategy

Evolution Strategy based ML





x-XOR PUF.

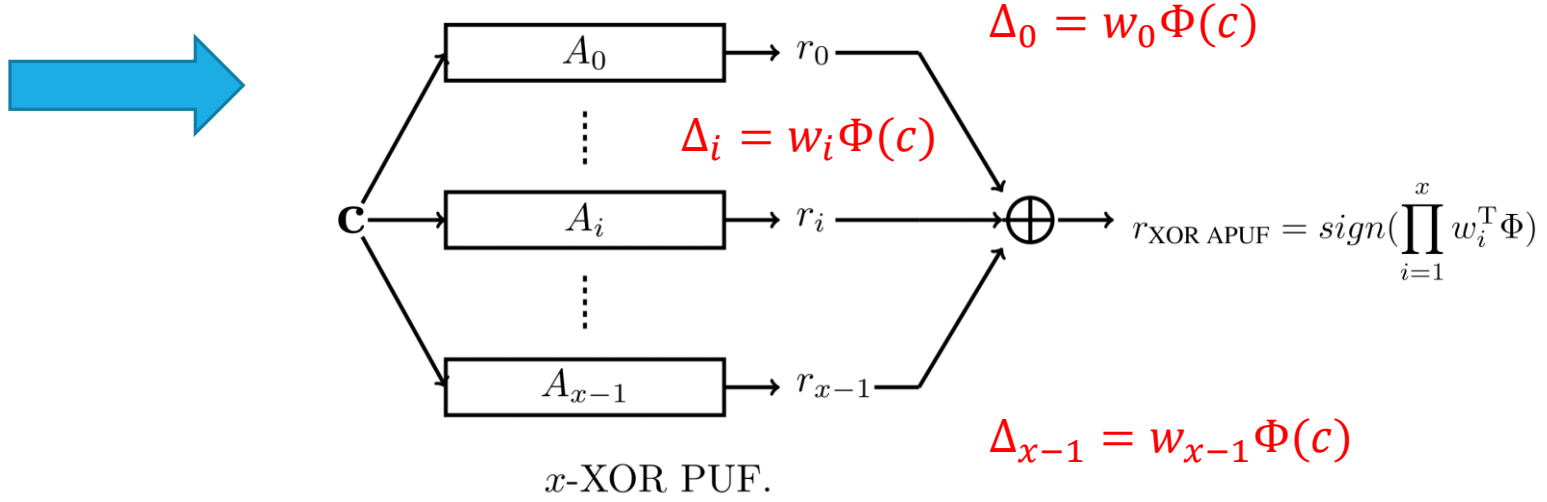
1. Use the precise model of XOR APUF
2. Not use gradient for optimization process.

Hence, it is not an optimal method for finding $\widehat{w}$

Attacks on x-XOR APUF: Logistic Regression

1. Evolution Strategy based ML

Logistic Regression



Attacks on x-XOR APUF: Logistic Regression

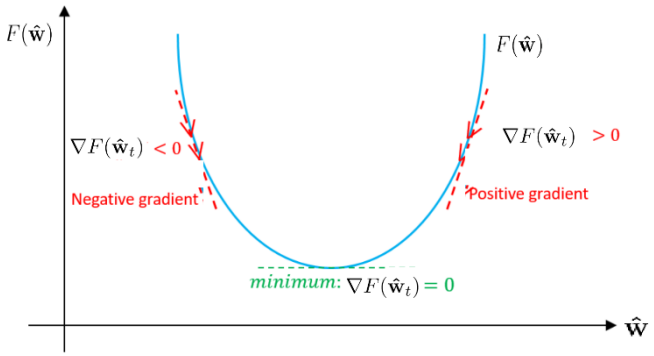
1. Evolution Strategy based ML

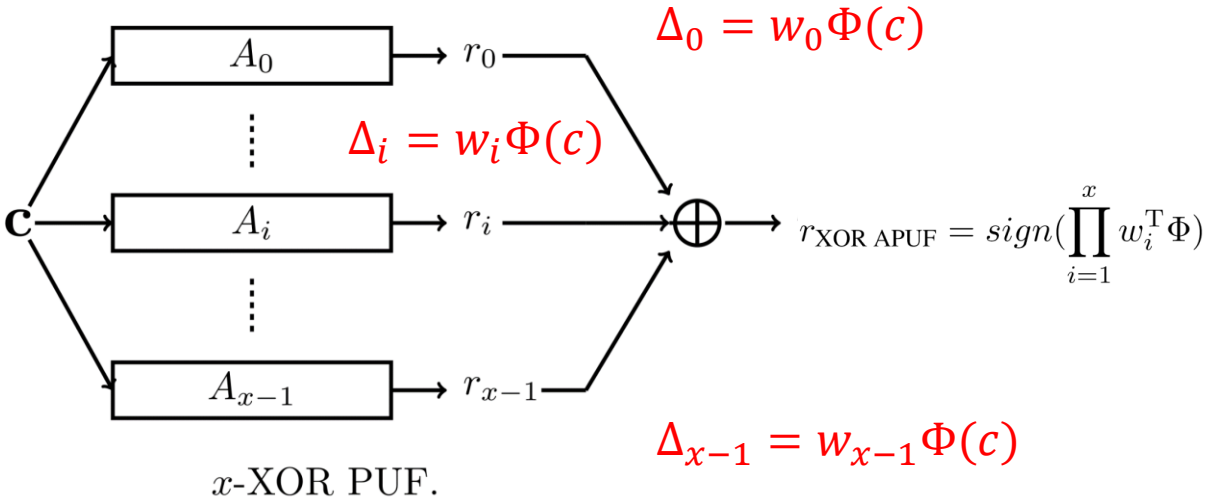
Logistic Regression

$$r_{\text{XOR APUF}} = sign(\prod_{i=1}^{x} w_i \Phi(\mathbf{c}))$$

$$\approx \sigma(\prod_{i=1}^{x} w_i \Phi(\mathbf{c}))$$

$$\sigma(x) = \frac{1}{1+\exp(-x)}$$





x-XOR PUF.

Attacks on x-XOR APUF: Logistic Regression

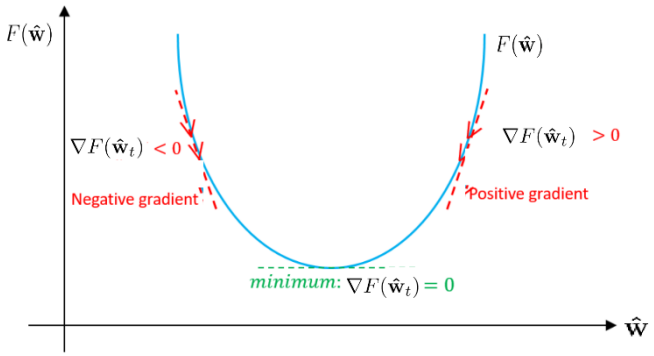
1. Evolution Strategy based ML

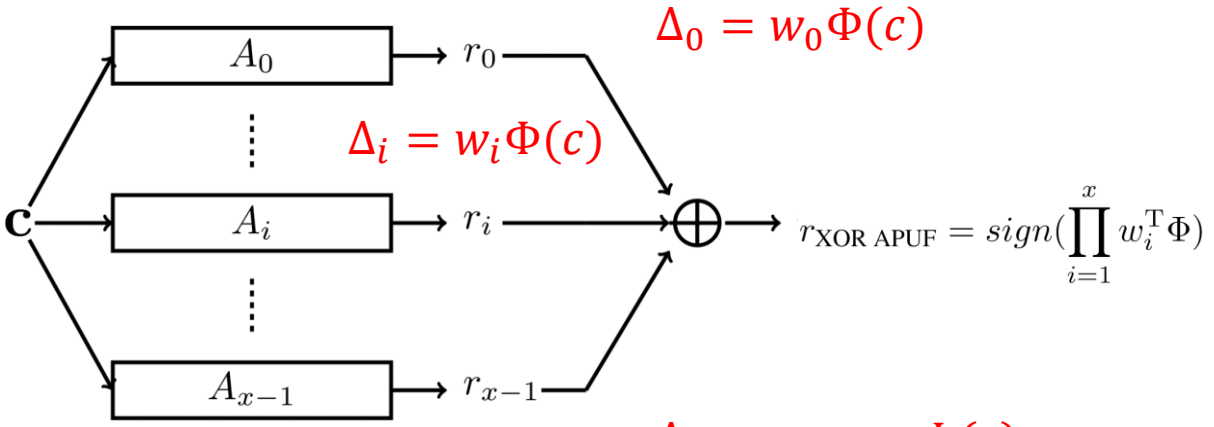
Logistic Regression

$$r_{\text{XOR APUF}} = sign(\prod_{i=1}^{x} w_i \Phi(\mathbf{c}))$$

$$\approx \sigma(\prod_{i=1}^{x} w_i \Phi(\mathbf{c}))$$

$$\sigma(x) = \frac{1}{1+\exp(-x)}$$





x-XOR PUF.

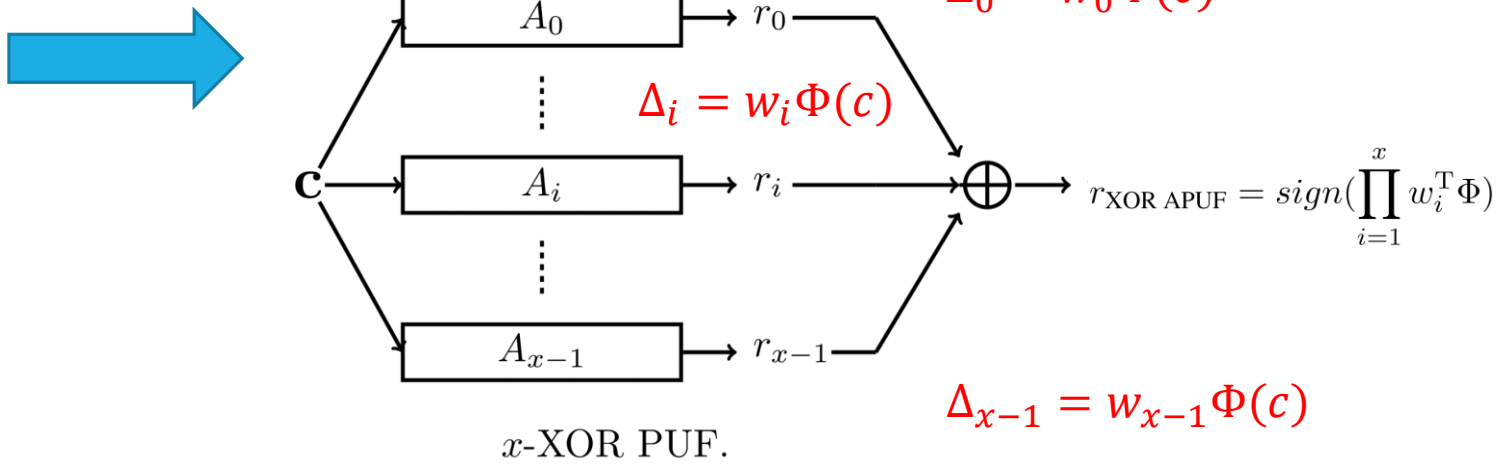
1. Use the **precise** model of XOR APUF
2. **Use gradient** for optimization process.

Hence, it is **an optimal** method for finding $\widehat{w}$

Attacks on x-XOR APUF: Deep Neural Network

1. **Evolution Strategy based ML**
2. **Logistic Regression based ML**

Deep Neural Network

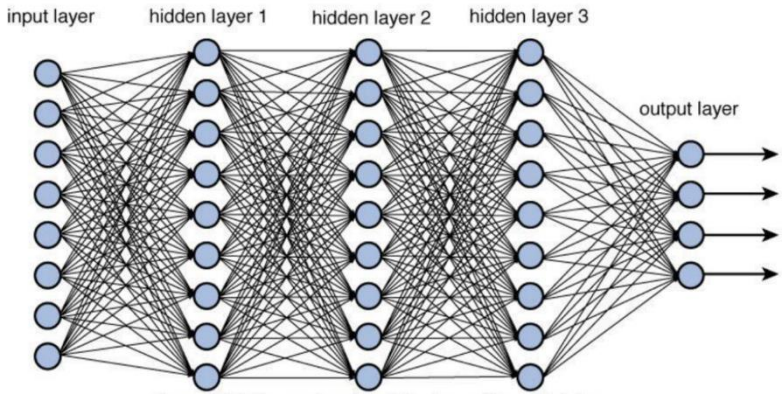


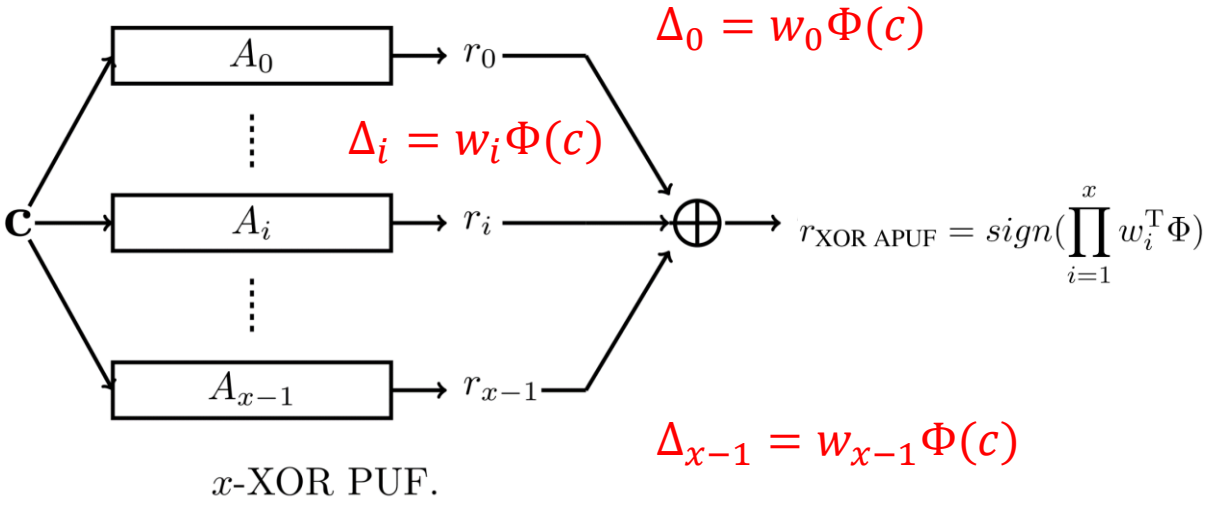
x-XOR PUF.

Attacks on x-XOR APUF: Deep Neural Network

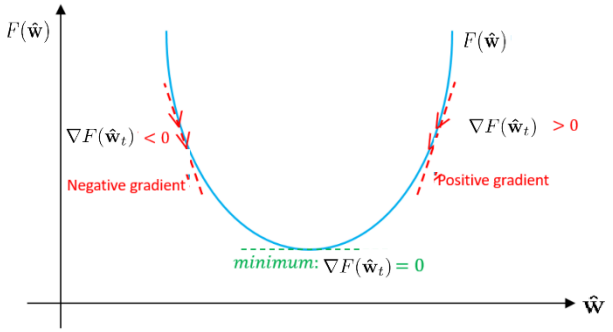
1. **Evolution Strategy based ML**
2. **Logistic Regression based ML**

Deep Neural Network





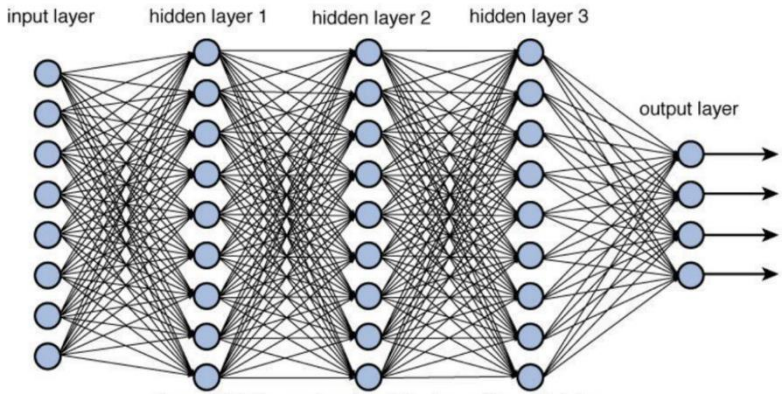
$$r_{\text{XOR APUF}} = sign(\prod_{i=1}^{x} w_i\Phi(\mathbf{c})) \approx \text{DNN}_{\mathbf{w}}(\Phi(\mathbf{c}))$$

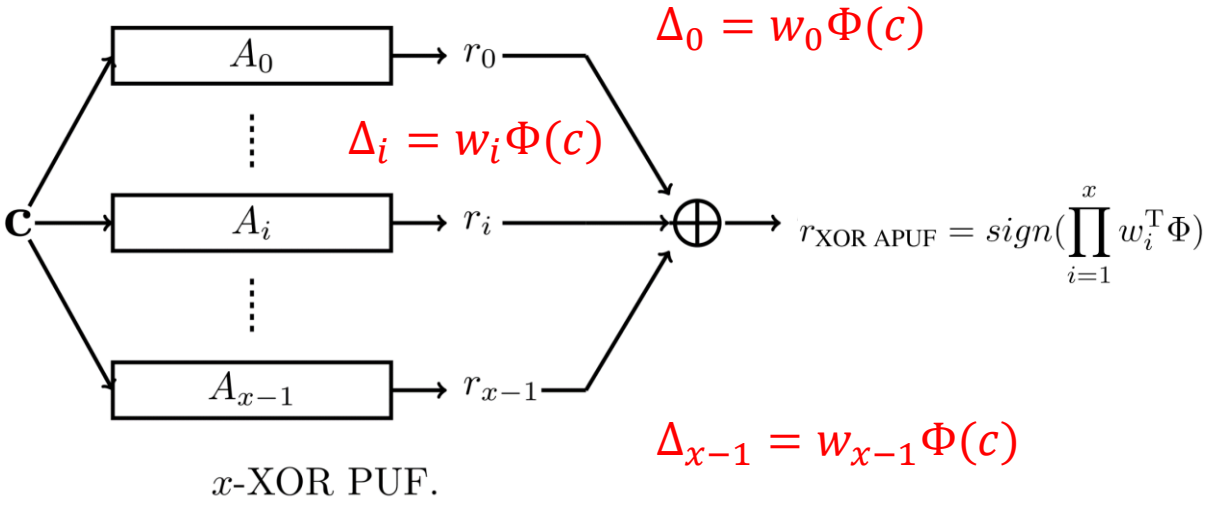


Attacks on x-XOR APUF: Deep Neural Network

1. **Evolution Strategy based ML**
2. **Logistic Regression based ML**

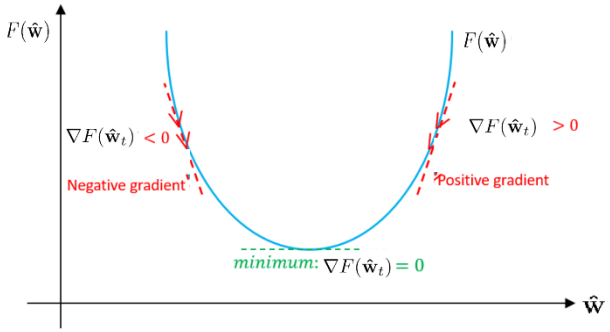
Deep Neural Network





x-XOR PUF.

$$r_{\text{XOR APUF}} = sign(\prod_{i=1}^{x} w_i\Phi(\mathbf{c})) \approx \text{DNN}_{\mathbf{w}}(\Phi(\mathbf{c}))$$

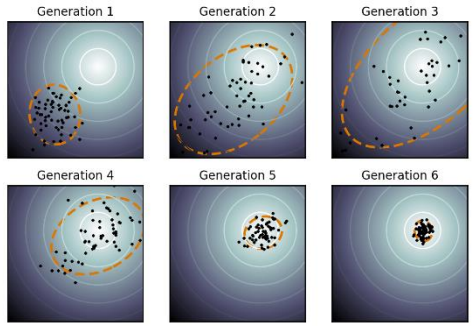


1. **Use non-precise model of XOR APUF**
2. **Use gradient for optimization process.**

Hence, it is not optimal method for finding $\hat{w}$

Attacks on x-XOR APUF: Comparison

Evolution Strategy based ML

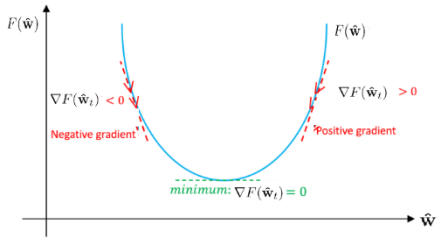


$$r_{\text{XOR APUF}} = sign(\prod_{i=1}^{x} w_i^{\mathrm{T}}\Phi)$$

1. Use precise model of XOR APUF
2. Not use gradient for optimization process.

Hence, it is not optimal method for finding $\widehat{w}$

Logistic Regression based ML

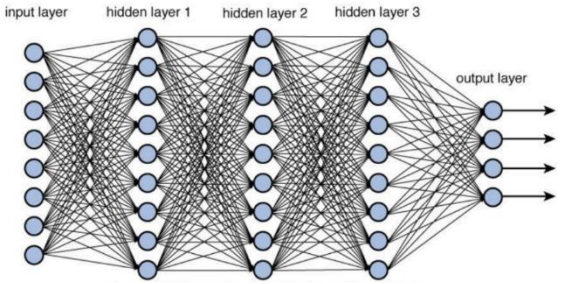


$$r_{\text{XOR APUF}} = sign(\prod_{i=1}^{x} w_i^{\mathrm{T}}\Phi)$$

1. Use precise model of XOR APUF
2. Use gradient for optimization process.

Hence, it is an optimal method for finding $\widehat{w}$

Deep Neural Network based ML



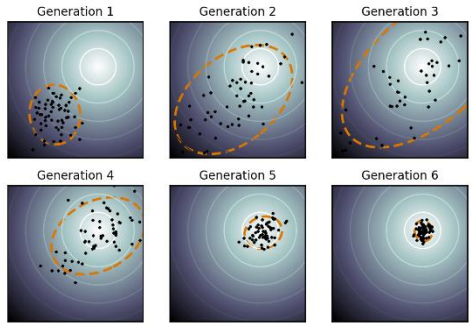
$$r_{\text{XOR APUF}} = sign(\prod_{i=1}^{x} w_i\Phi(\mathbf{c}))$$
$$\approx \mathrm{DNN}_{\mathbf{w}}(\Phi(\mathbf{c}))$$

1. Use non-precise model of XOR APUF
2. Use gradient for optimization process.

Hence, it is not optimal method for finding $\widehat{w}$

Attacks on x-XOR APUF: Comparison

Evolution Strategy based ML

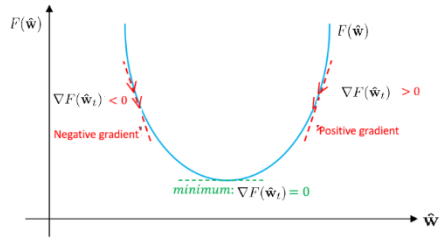


$$r_{\text{XOR APUF}} = sign(\prod_{i=1}^{x} w_i^{\mathrm{T}}\Phi)$$

1. Use precise model of XOR APUF
2. Not use gradient for optimization process.

Hence, it is not optimal method for finding $\widehat{w}$

Logistic Regression based ML



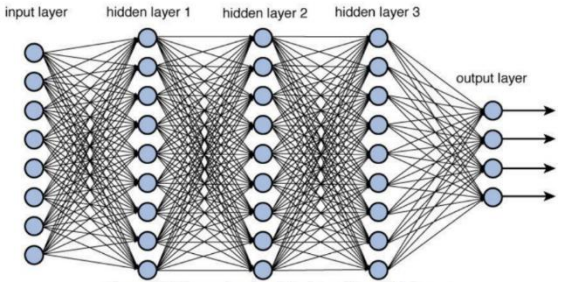
$$r_{\text{XOR APUF}} = sign(\prod_{i=1}^{x} w_i^{\mathrm{T}}\Phi)$$

1. Use precise model of XOR APUF
2. Use gradient for optimization process.

Hence, it is an optimal method for finding $\widehat{w}$

Best Attack

Deep Neural Network based ML



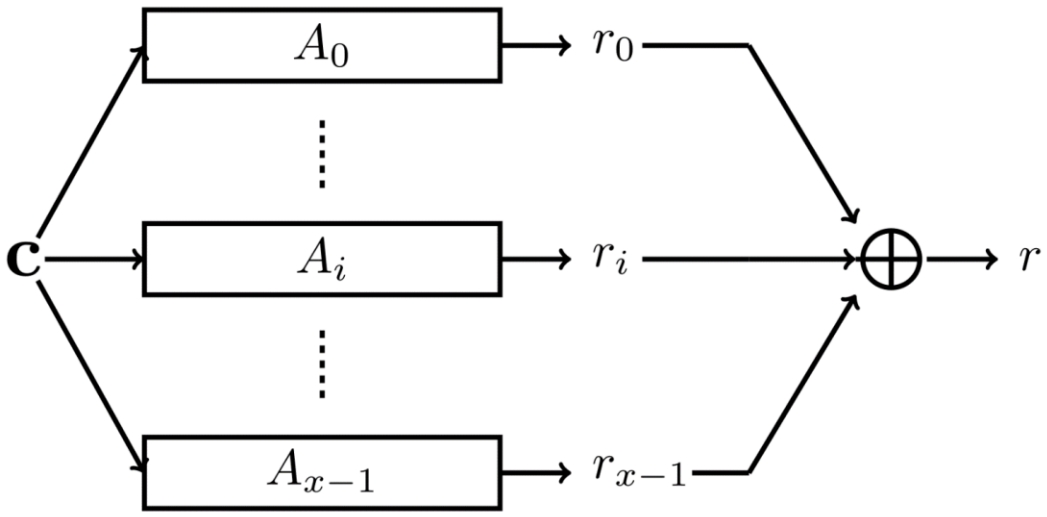
$$r_{\text{XOR APUF}} = sign(\prod_{i=1}^{x} w_i\Phi(\mathbf{c}))$$
$$\approx \mathrm{DNN}_{\mathbf{w}}(\Phi(\mathbf{c}))$$

1. Use non-precise model of XOR APUF
2. Use gradient for optimization process.

Hence, it is not optimal method for finding $\widehat{w}$

UCONN

x-XOR APUF



x-XOR PUF.

Security: when the number of APUFs is large enough (64-bit 10-XOR APUF), then the CRP-based modeling attacks are not practical anymore.

As shown in [6], in case of Logistic Regression based modelling attacks on n-bit x-XOR PUF has , the number of iterations for the training phase is: $N_{iteration} = O(\frac{(n+1)^x}{N_{data} \cdot x!})$ and the total time complexity is:

$$T = N_{iteration} \cdot (n+1) \cdot x \cdot N_{data}$$

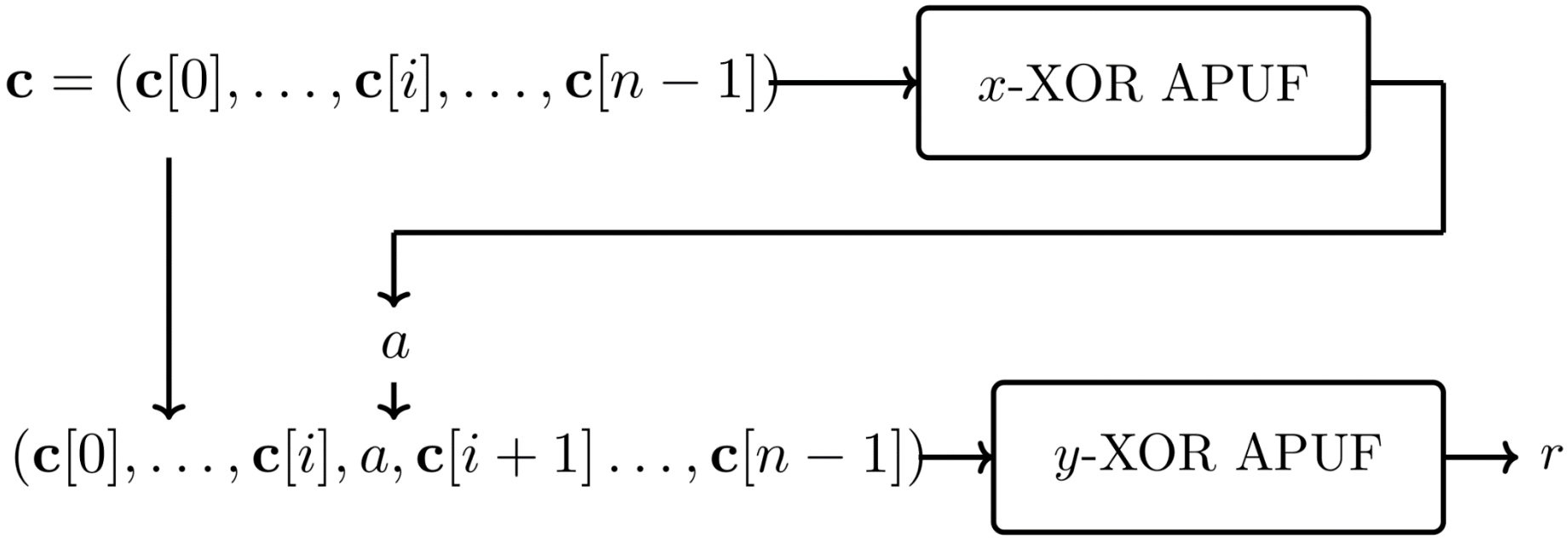
Physical unclonable functions for device authentication and secret key generation. DAC2007, G. E. Suh and S. Devadas

UCONN

6. Interpose PUF (iPUF) and Classical Modeling Attacks

UCONN

5. Interposed PUF (iPUF)

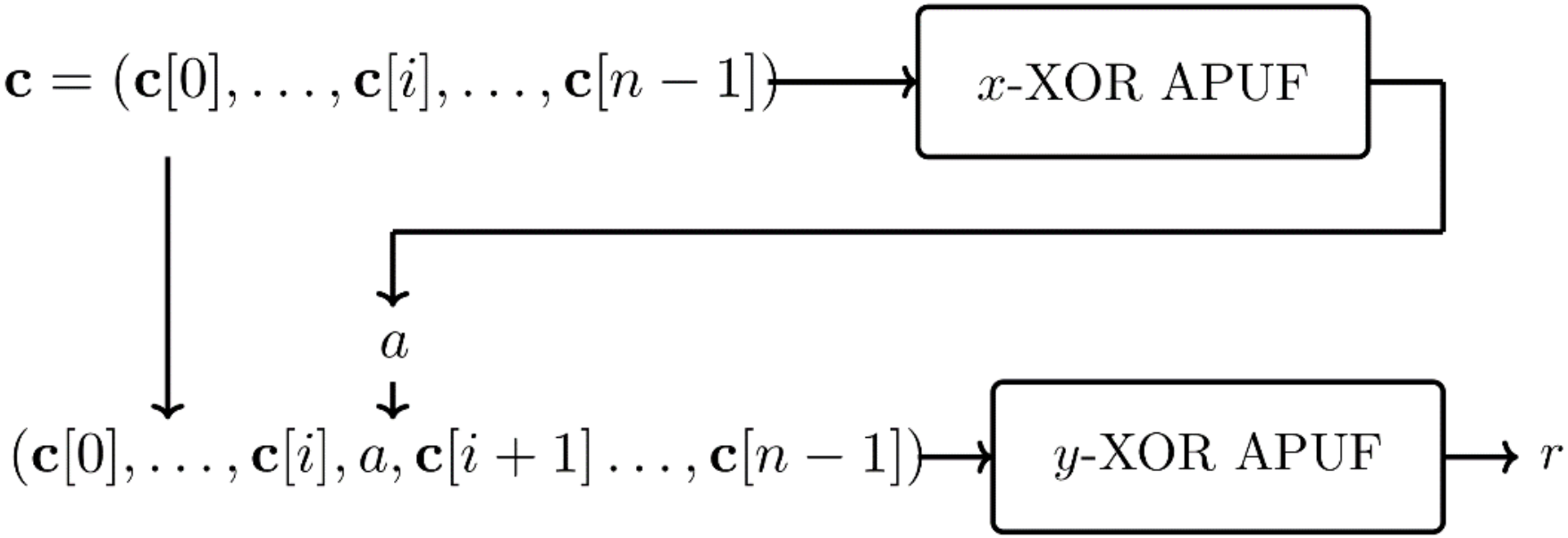


Security of iPUF wrt CRP-based Modeling attacks

Traditional Modeling Attacks

$$(x, y) - IPUF \approx \left(y + \frac{x}{2}\right) - XOR\ PUF$$

if a is inserted at the middle



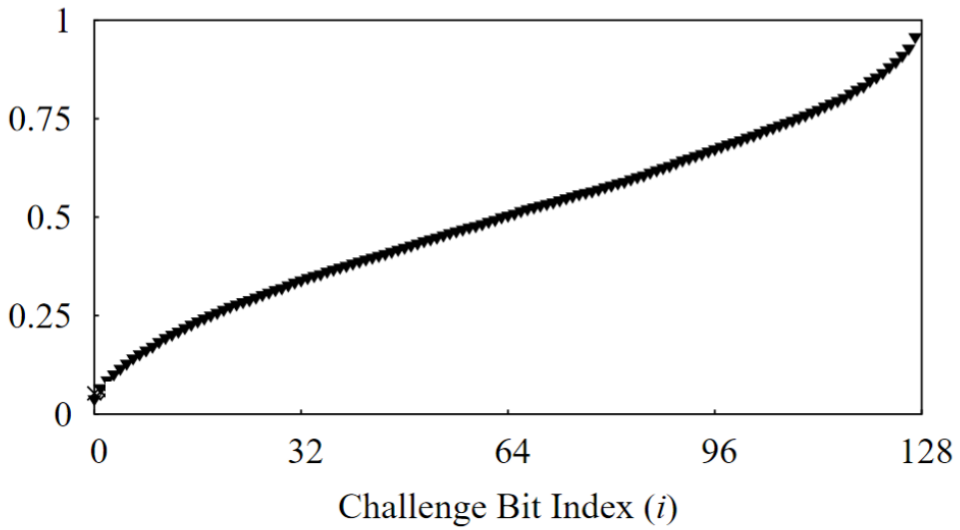
UCONN

Road Map

- Strict Avalanche Criterion (SAC) Property of APUF
- Influence of each APUF in XOR APUF

Strict Avalanche Criterion (SAC) Property

Strict Avalanche Criterion (SAC) Property



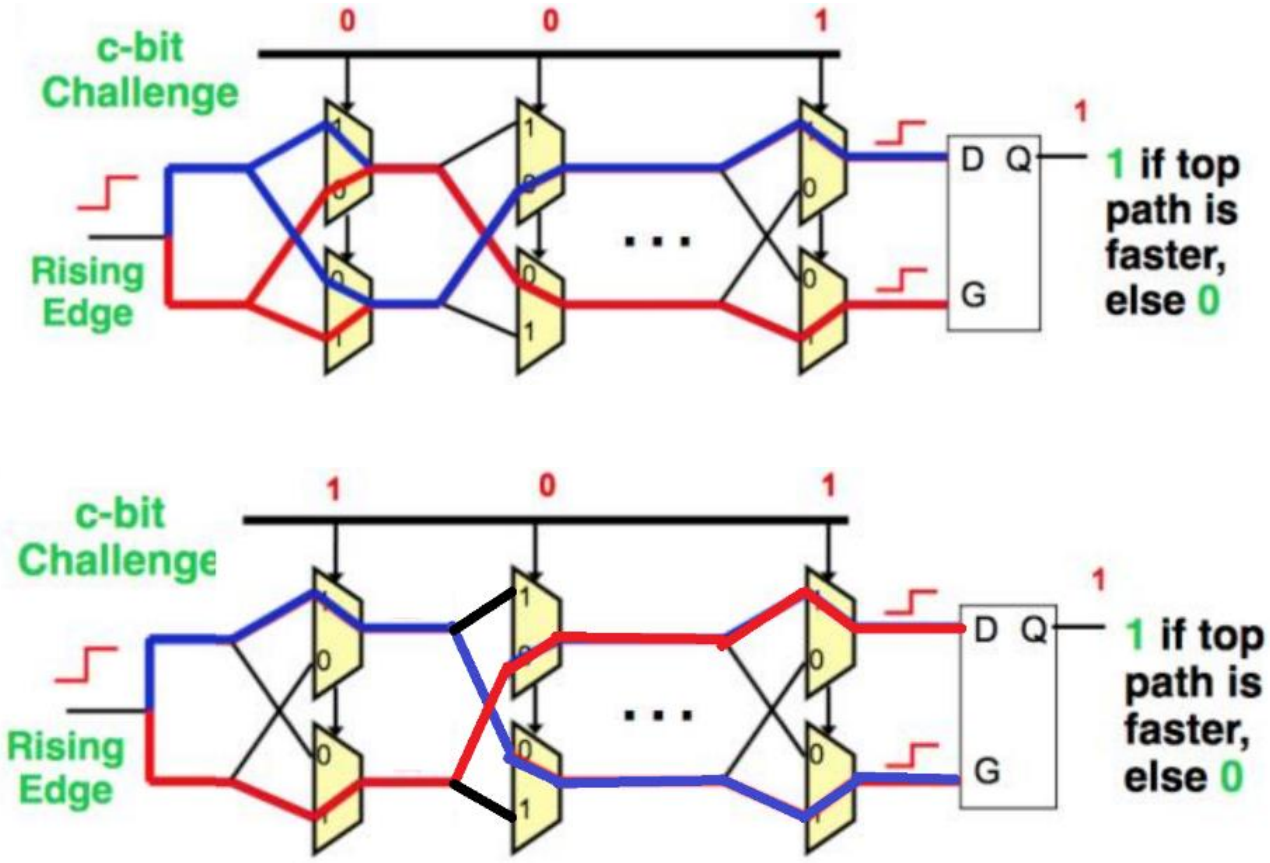
(b) 128-bit APUF

$$Pr_{\mathbf{c}}(r_0 \neq r_1) \approx \frac{n-i}{n}$$

$$Pr_{\mathbf{c}}(r_0 \neq r_1) = \frac{1}{2},\ i = \frac{n}{2}$$

UCONN

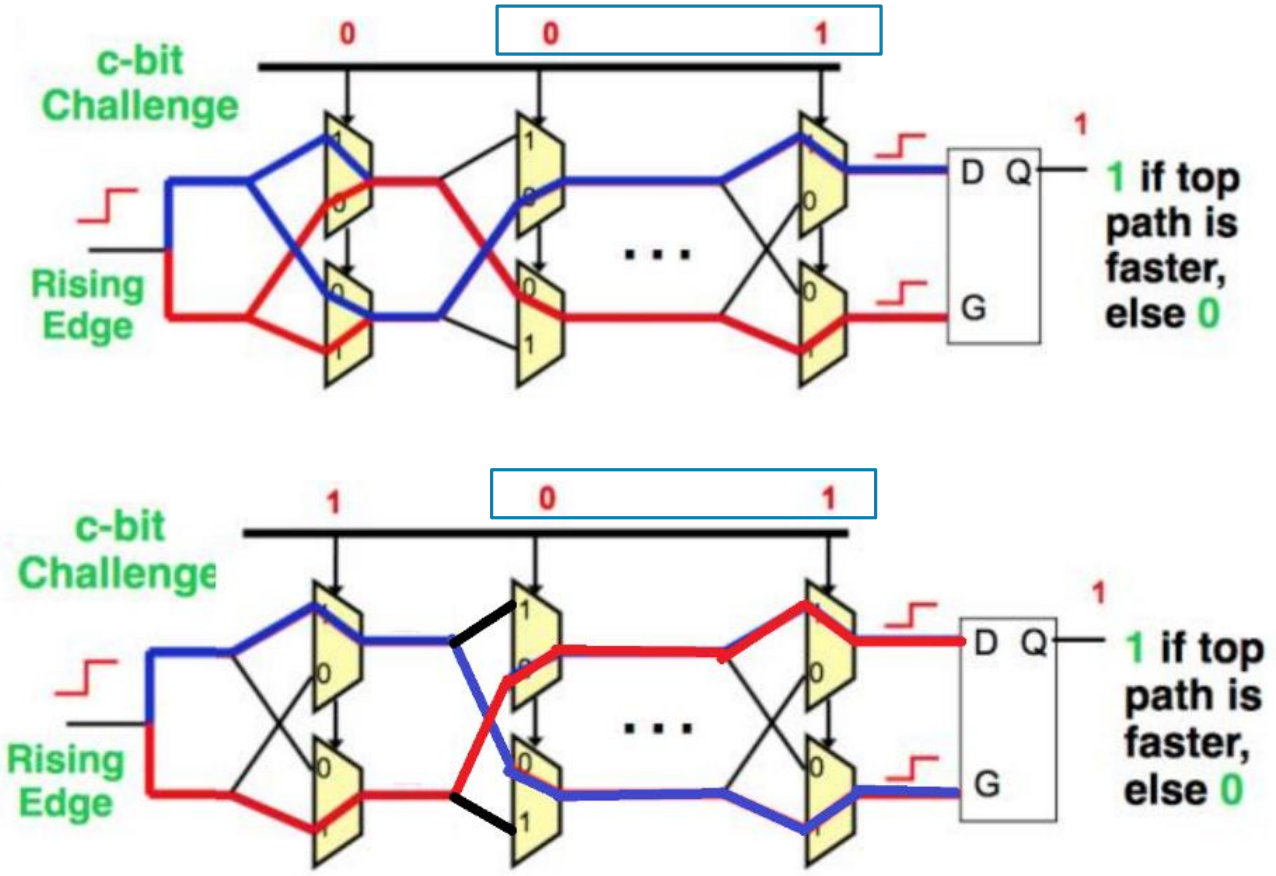
Position of Challenge bit of APUF



Flipping c[0]

UCONN

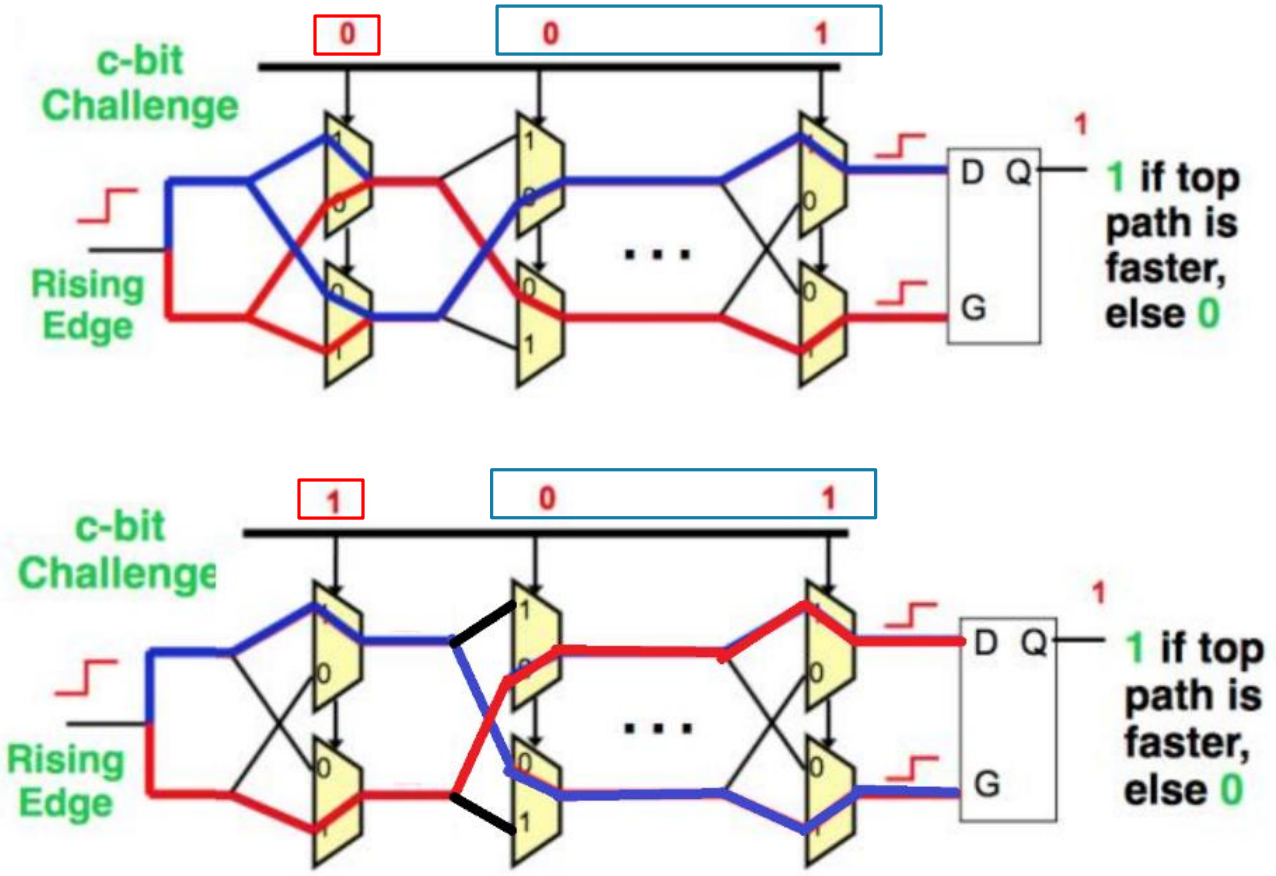
Position of Challenge bit of APUF



Flipping c[0]

UCONN

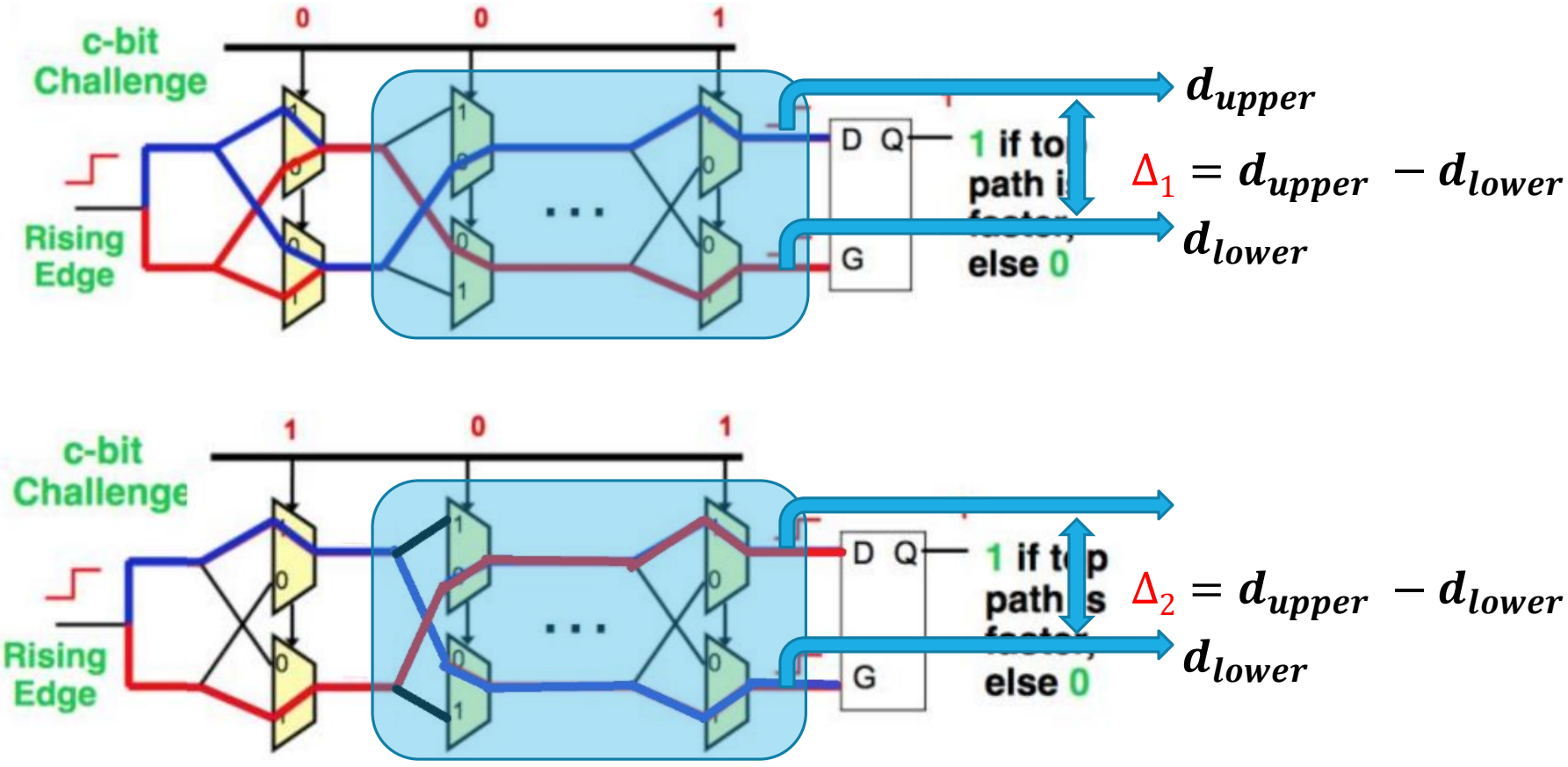
Position of Challenge bit of APUF



Flipping c[0]

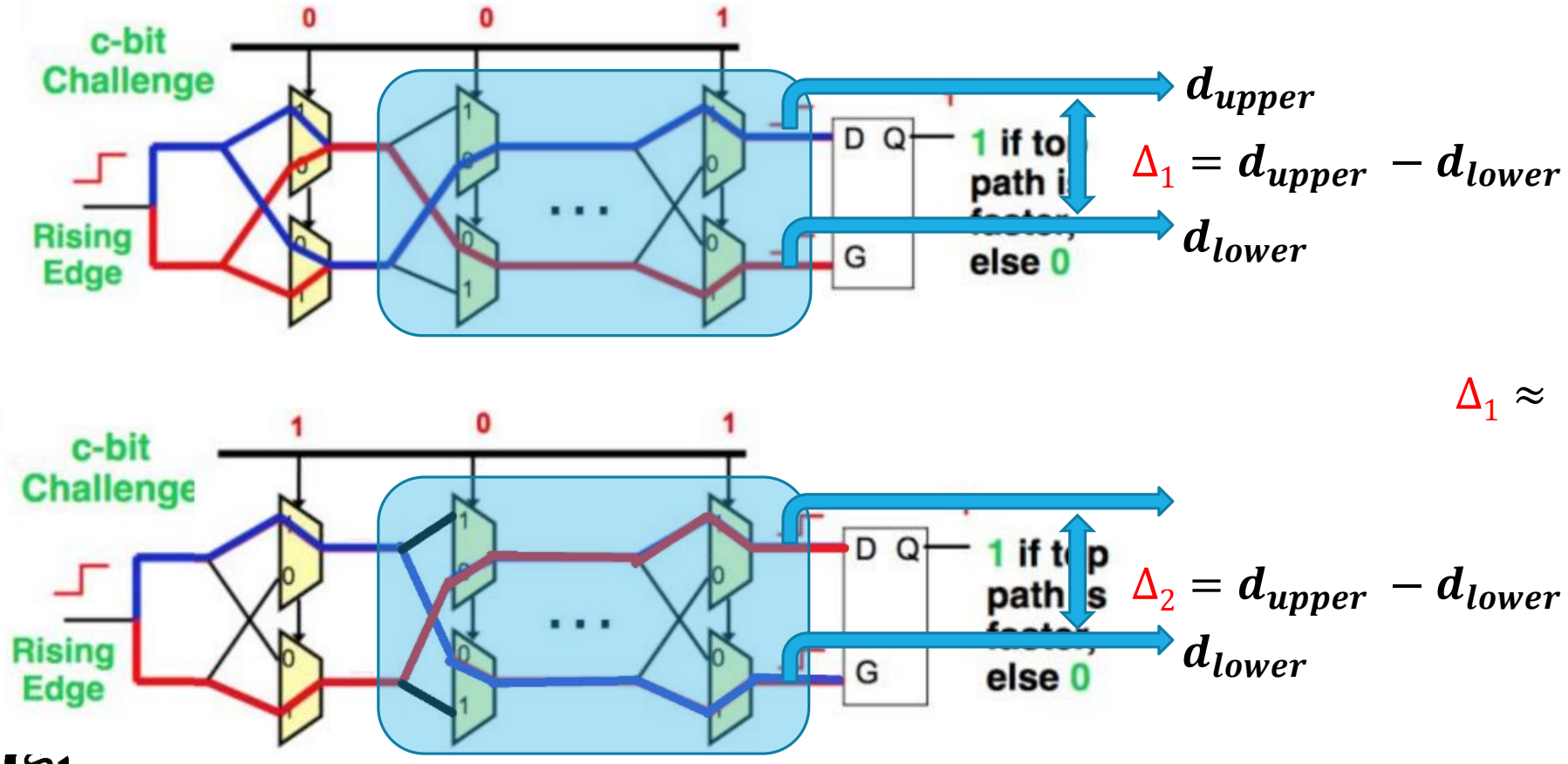
UCONN

Position of Challenge bit of APUF



Flipping c[0]

Position of Challenge bit of APUF

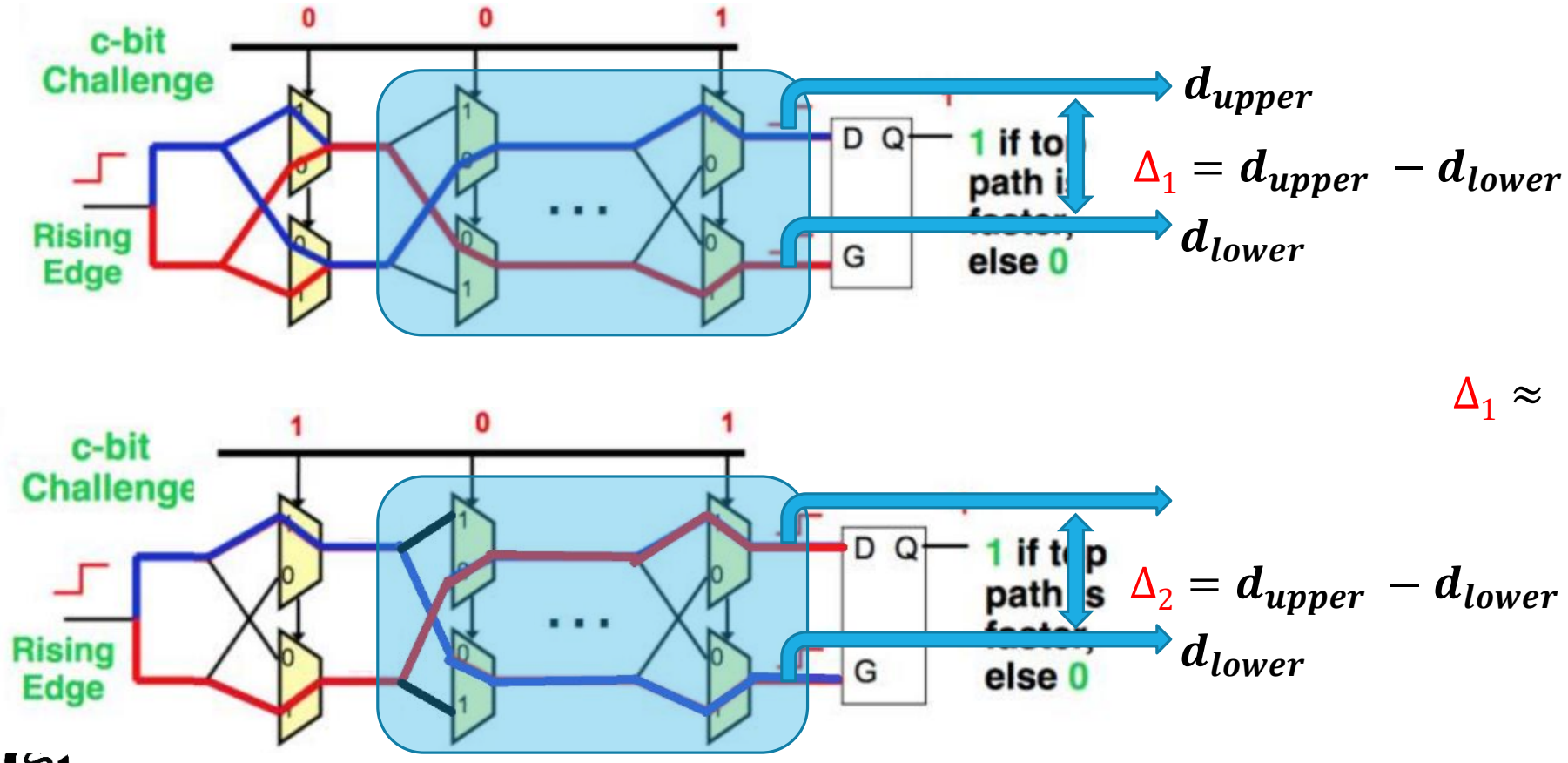


$\Delta_1 \approx \Delta_2$

Flipping c[0]

UCONN

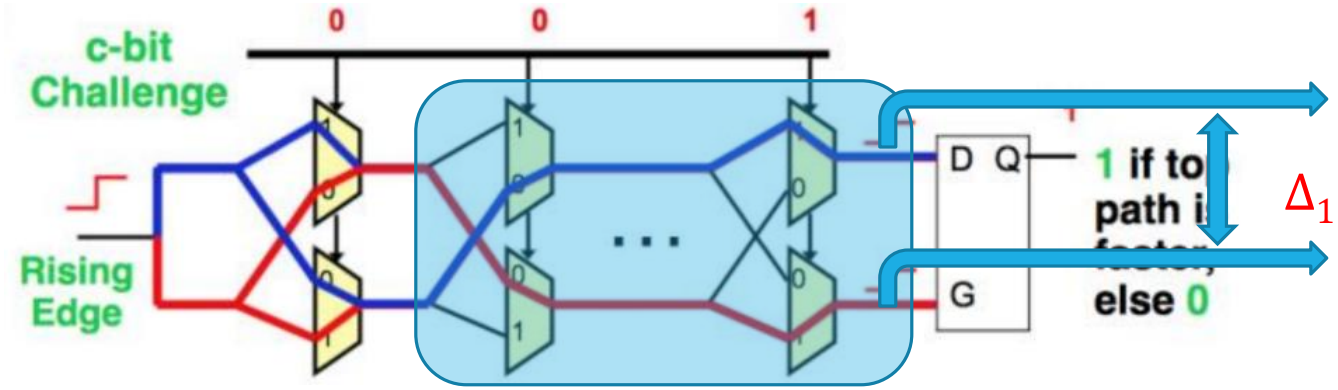
Position of Challenge bit of APUF



$\Delta_1 \approx \Delta_2 \rightarrow same\ responses$

Flipping c[0]

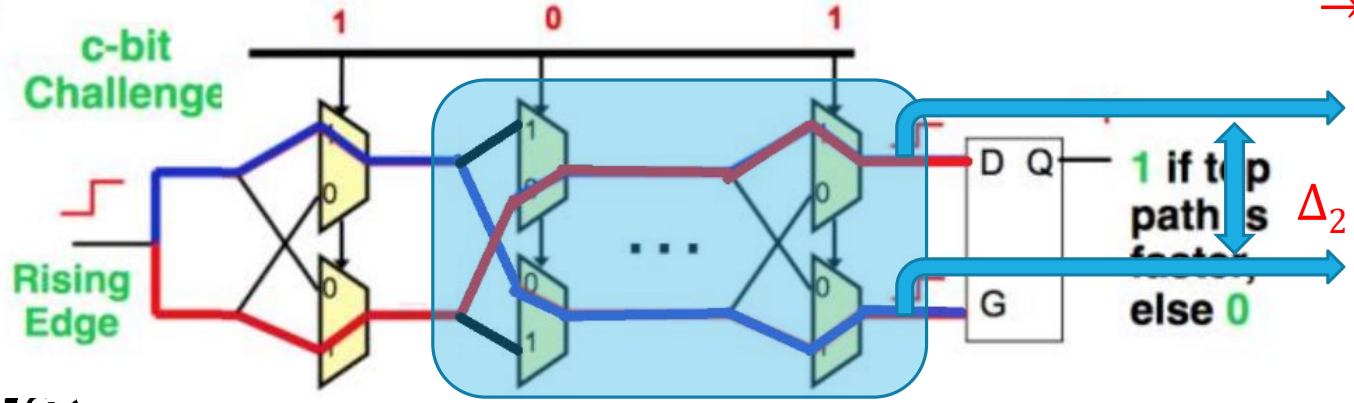
Position of Challenge bit of APUF



d_{upper}

$\Delta_1 = d_{upper} - d_{lower}$

d_{lower}



$\Delta_2 = d_{upper} - d_{lower}$

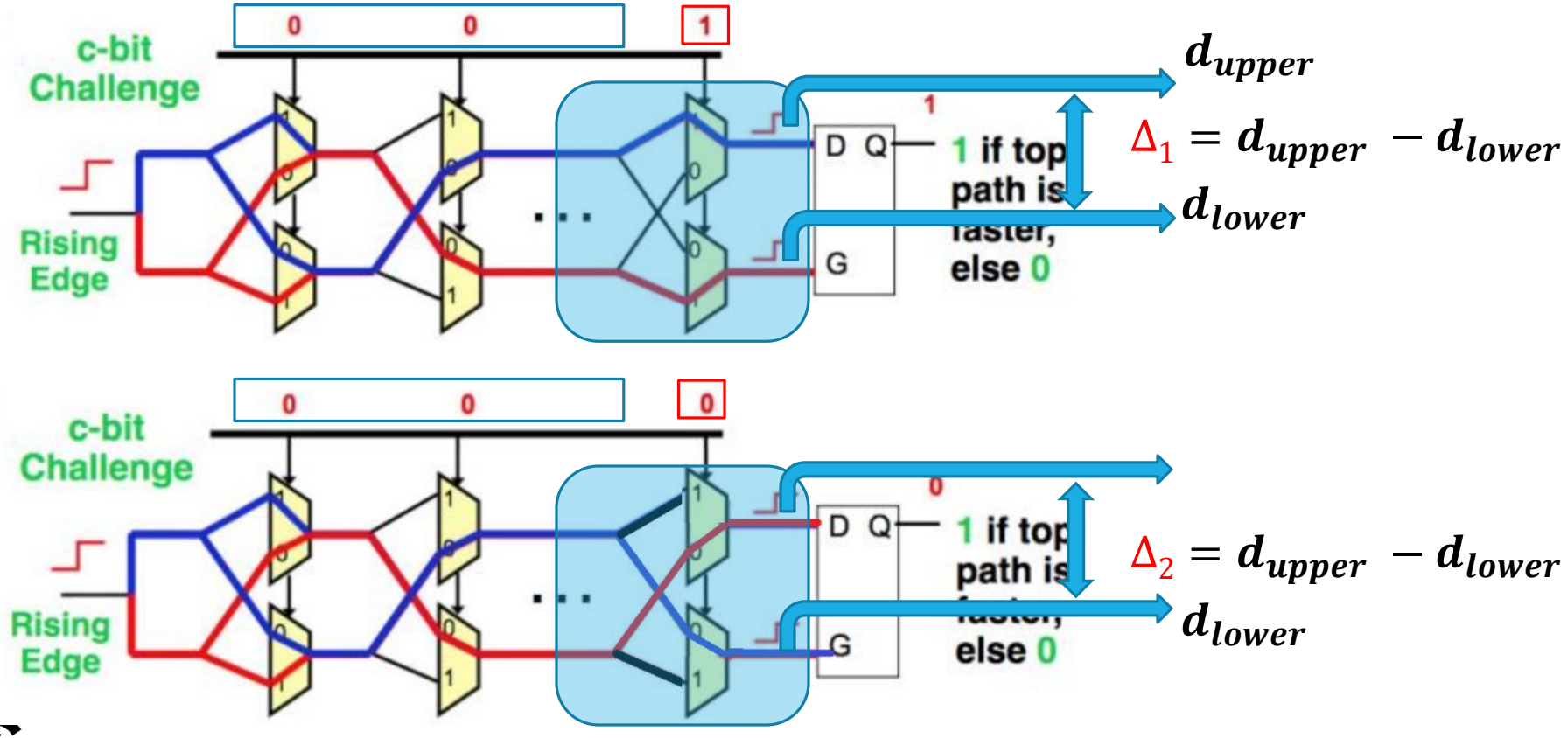
d_{lower}

$\Delta_1 \approx \Delta_2 \rightarrow same\ responses$

$\rightarrow c[0]\ has\ very\ less\ influence\ on\ APUF's\ output$

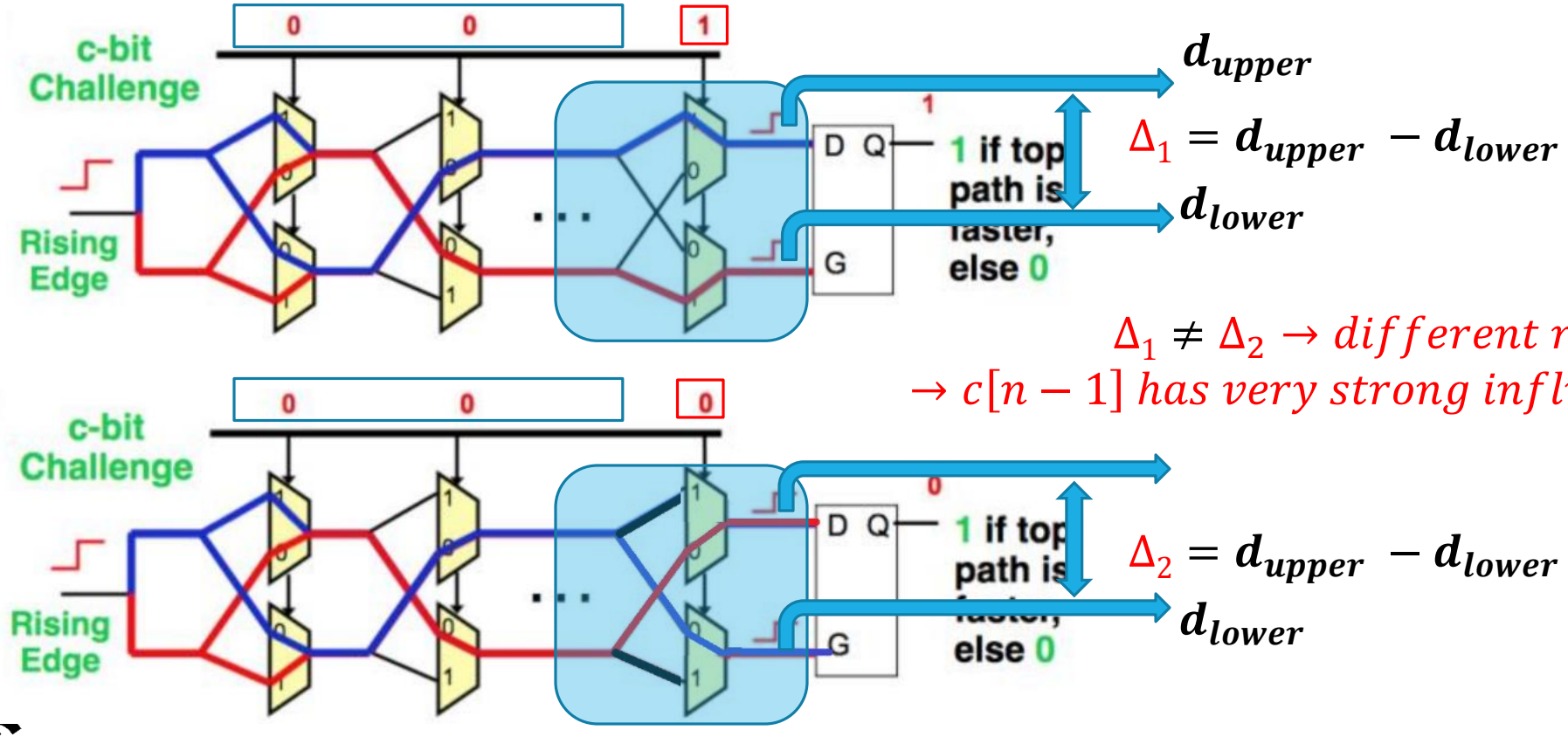
Flipping c[0]

Position of Challenge bit of APUF



Flipping c[n-1]

Position of Challenge bit of APUF

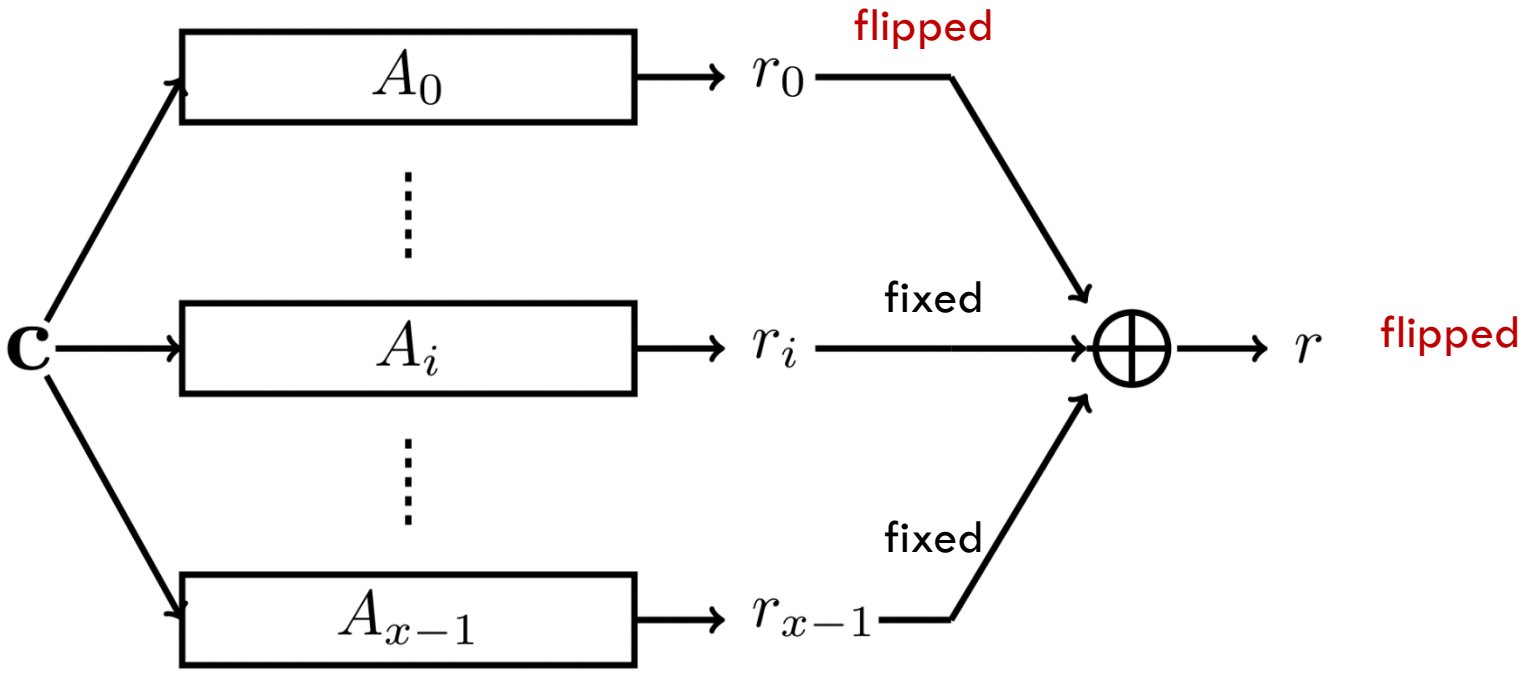


$\Delta_1 \neq \Delta_2 \rightarrow different\ responses$

$\rightarrow c[n-1]\ has\ very\ strong\ influence\ on\ APUF's\ output$

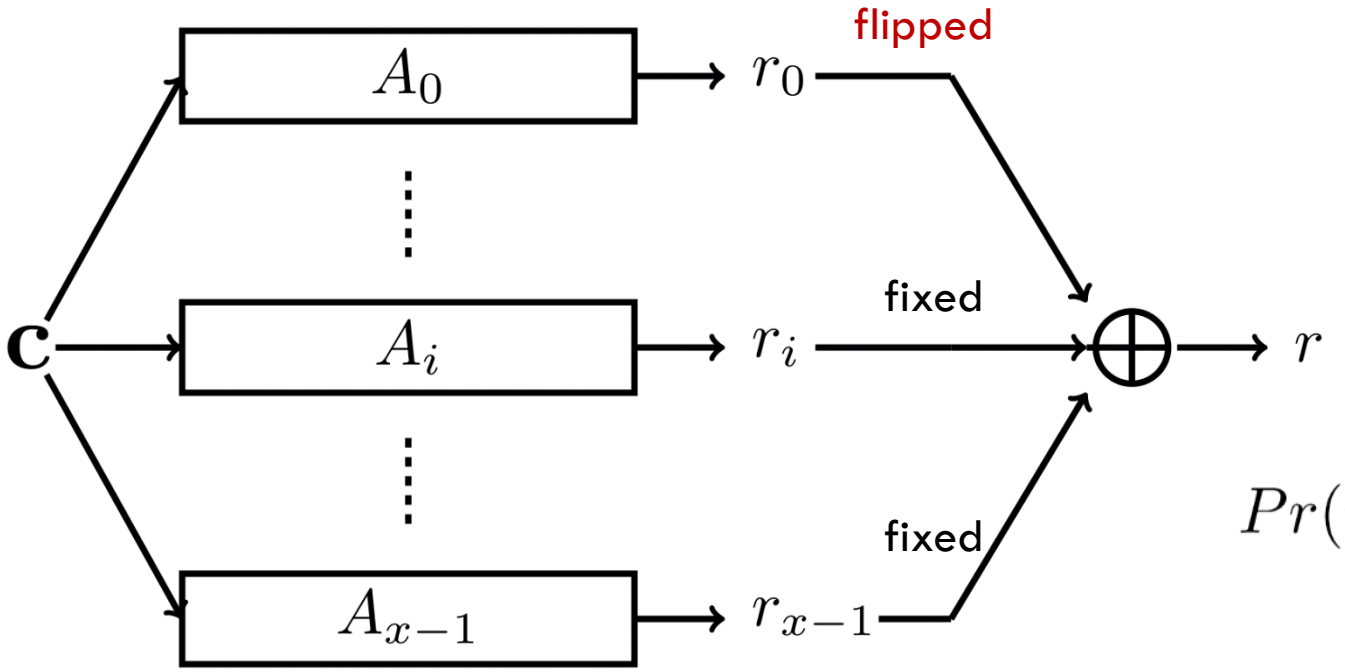
Flipping c[n-1]

Impact of APUFs' outputs on XOR APUF's output



x-XOR PUF.

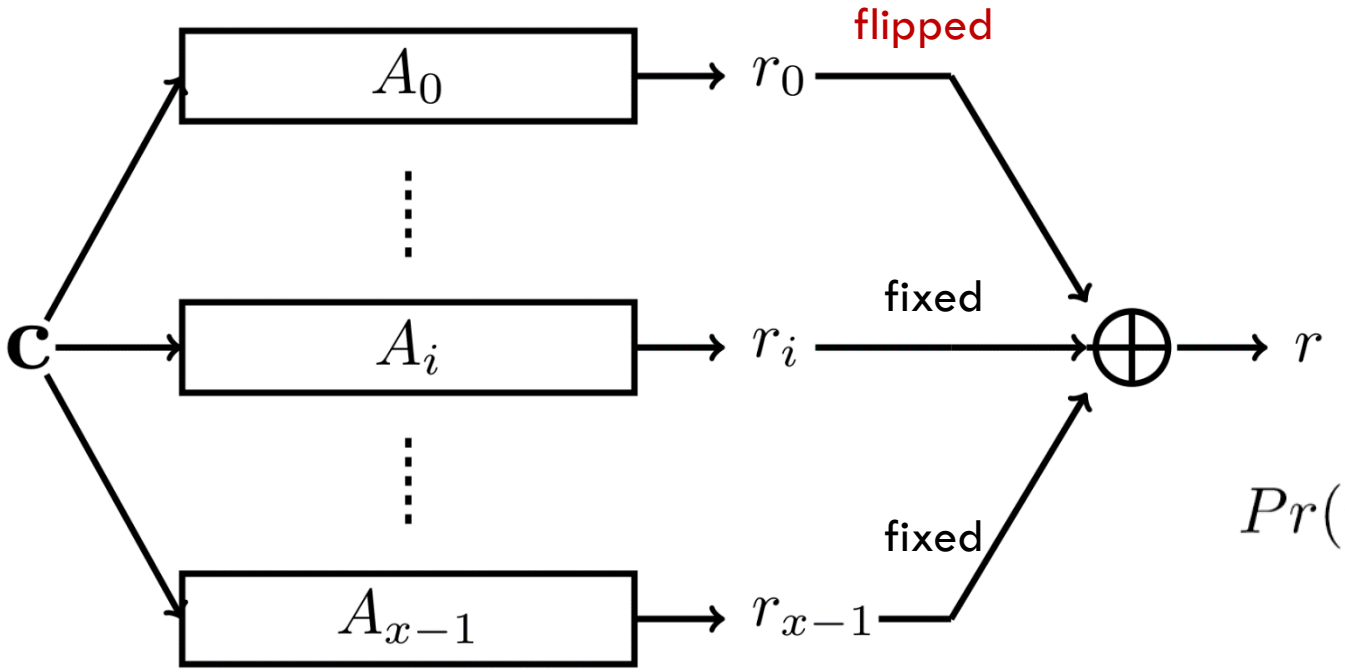
Impact of APUFs' outputs on XOR APUF's output



x-XOR PUF.

$$Pr(r \text{ flipped} | r_i \text{ flipped}, r_{j \neq i} \text{ fixed}) = 1$$

Impact of APUFs' outputs on XOR APUF's output



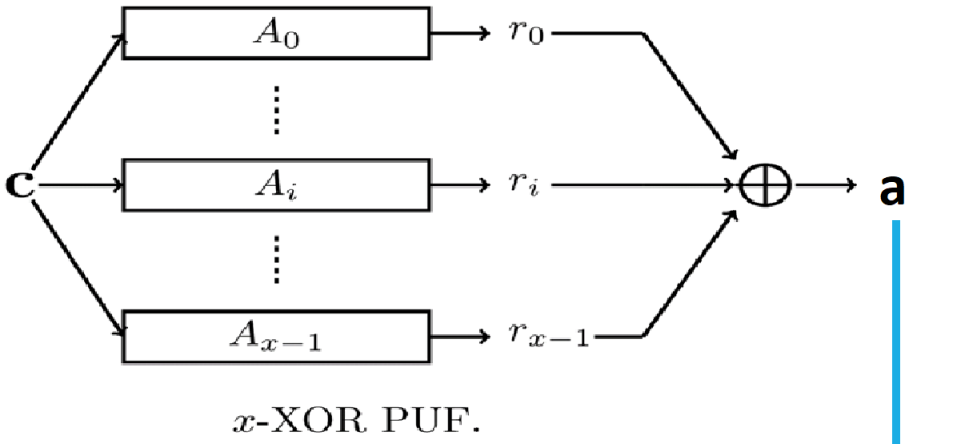
x-XOR PUF.

$$Pr(r \text{ flipped} | r_i \text{ flipped}, r_{j \neq i} \text{ fixed}) = 1$$

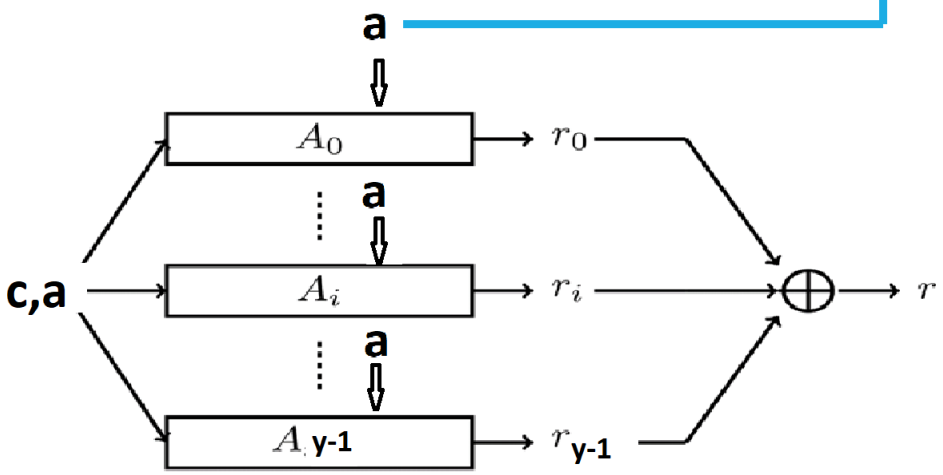
All APUFs' outputs contribute to the XOR APUF's output

XOR APUF and iPUF

(x,y)-iPUF



x-XOR PUF.

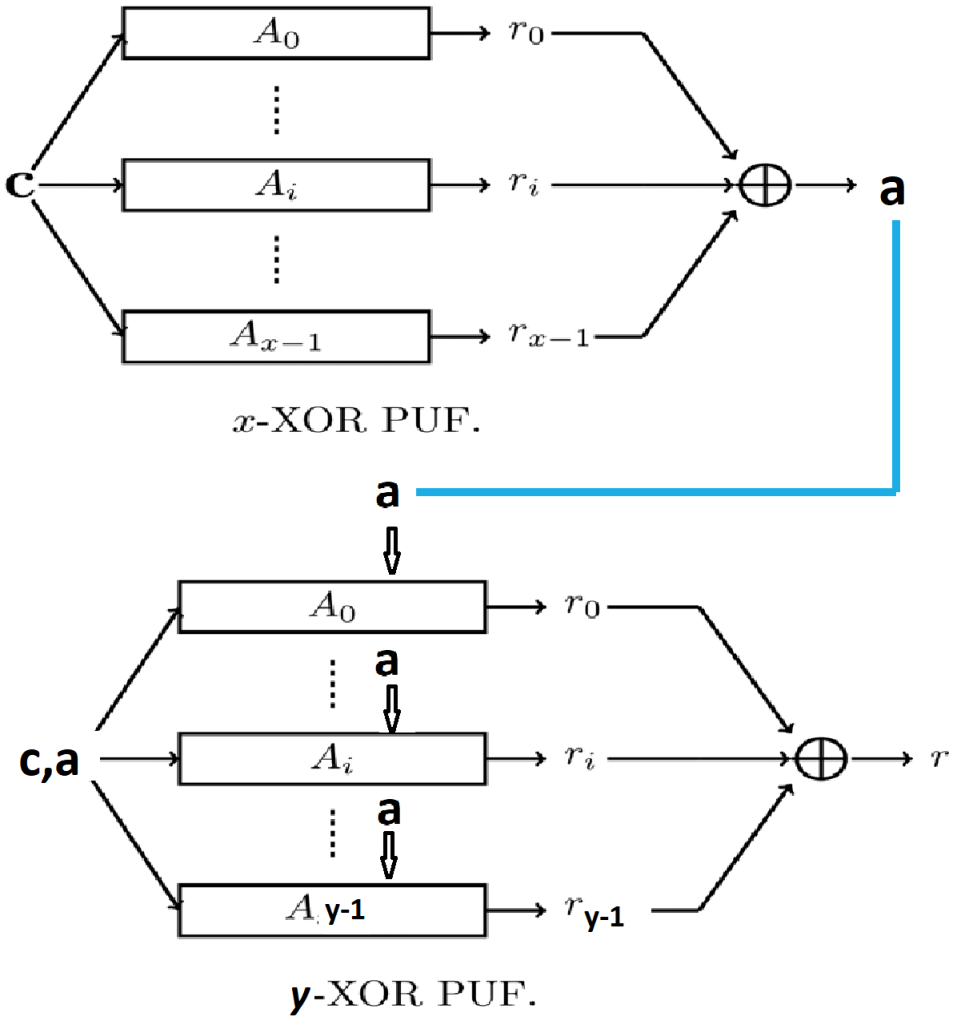


y-XOR PUF.

- All APUFs' outputs in x-XOR APUF contribute to a

XOR APUF and iPUF

(x,y)-iPUF

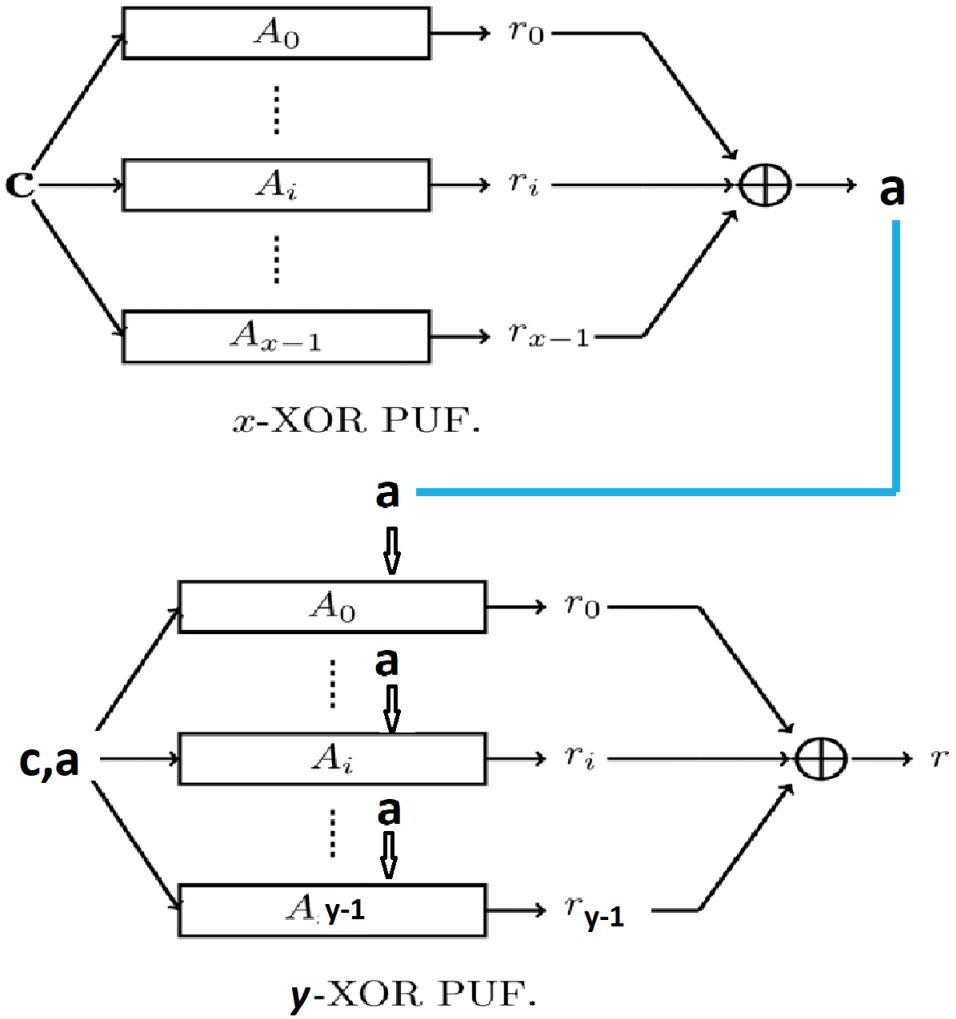


- All APUFs' outputs in x-XOR APUF contribute to a
- All APUF's outputs in y-XOR APUF contribute to r

UCONN

XOR APUF and iPUF

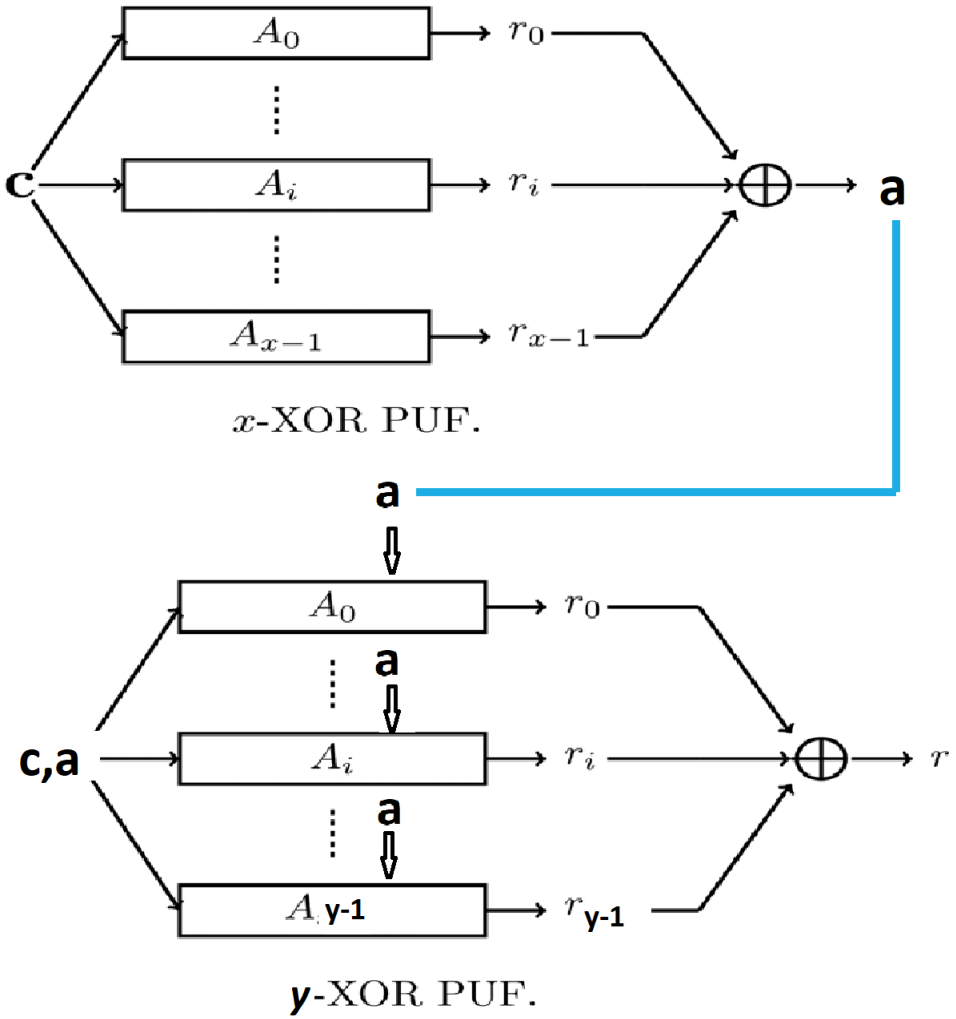
(x,y)-iPUF



- All APUFs' outputs in x-XOR APUF contribute to a
- All APUF's outputs in y-XOR APUF contribute to r
- For a given challenge c,
  If flipping a makes iPUF's output r not flip due to SAC of APUF,
  then (x,y)-iPUF = y-XOR APUF
  because only y APUFs at y-XOR APUF contribute to r.

XOR APUF and iPUF

(x,y)-iPUF

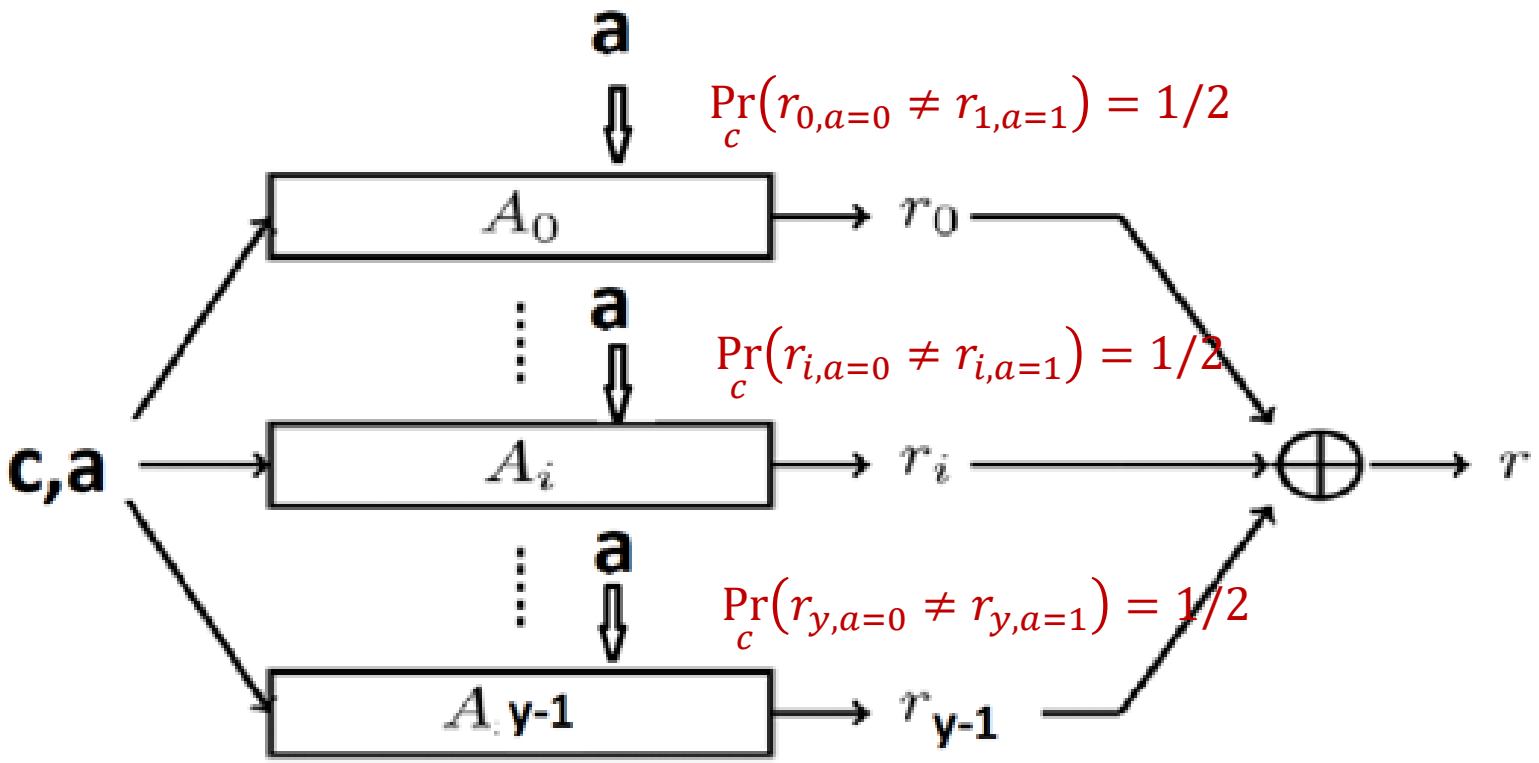


- All APUFs' outputs in x-XOR APUF contribute to a
- All APUF's outputs in y-XOR APUF contribute to r
- For a given challenge c,
  If flipping a makes iPUF's output r not flip due to SAC of APUF,
  then (x,y)-iPUF = y-XOR APUF
  because only y APUFs at y-XOR APUF contribute to r.

- For a given challenge c,
  If flipping a makes iPUF's output r flip due to SAC of APUF,
  then (x,y)-iPUF = (x+y)-XOR APUF
  because all x+y APUFs at x and y-XOR APUFs contribute to r

UCONN

XOR APUF and iPUF

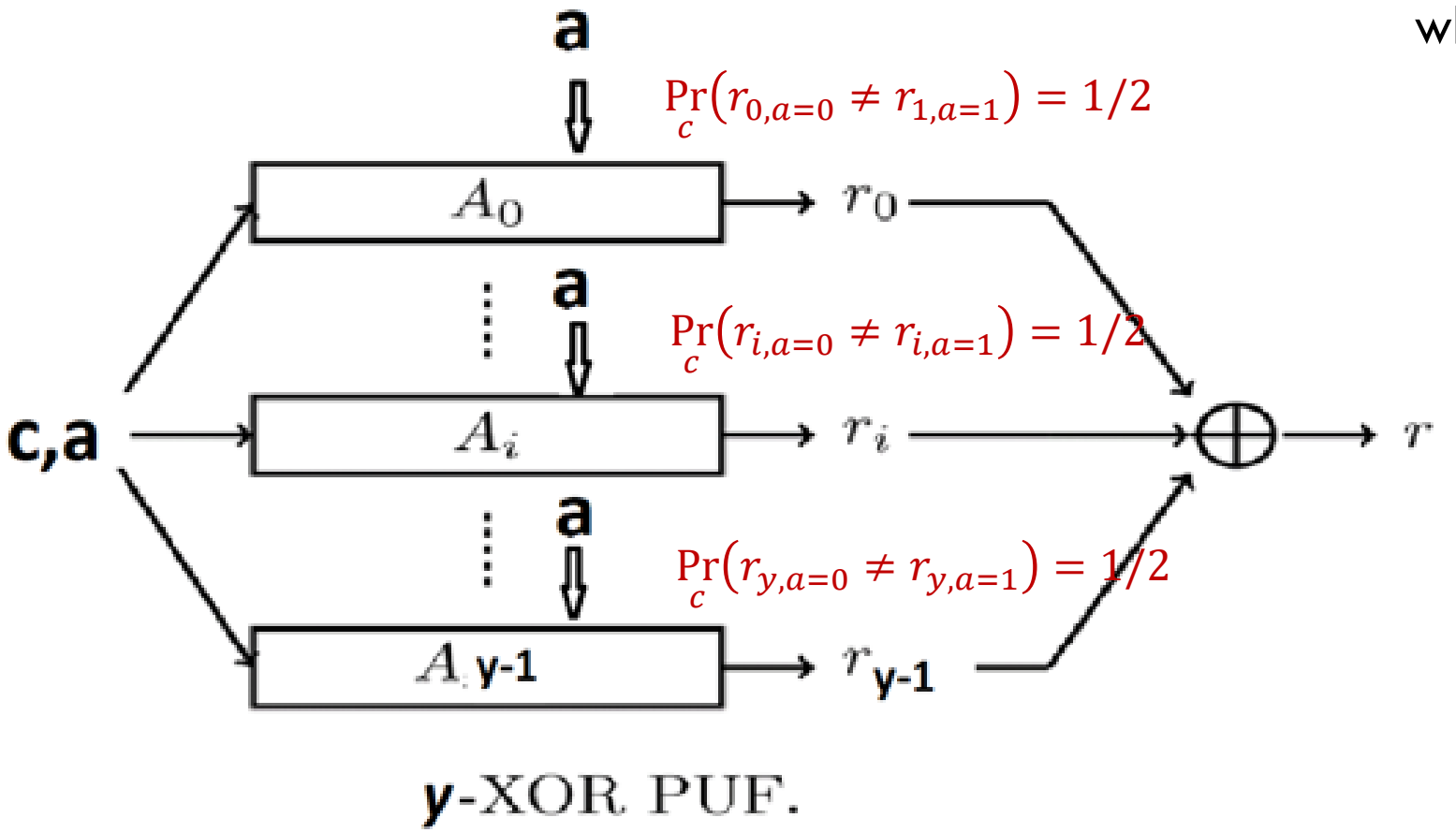


y-XOR PUF.

UCONN

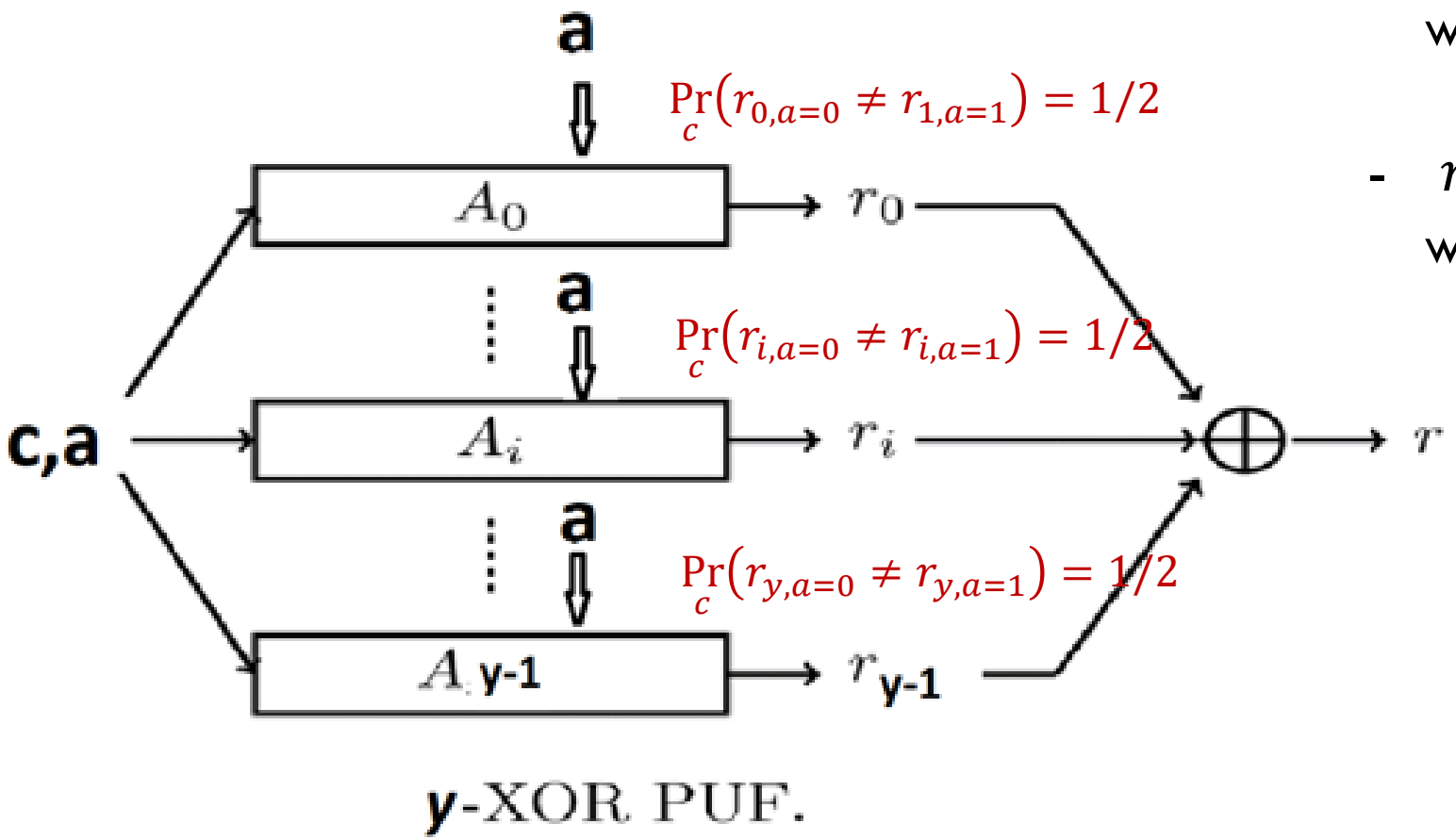
XOR APUF and iPUF

- r flips when odd # flipped r_i when a flips



y-XOR PUF.

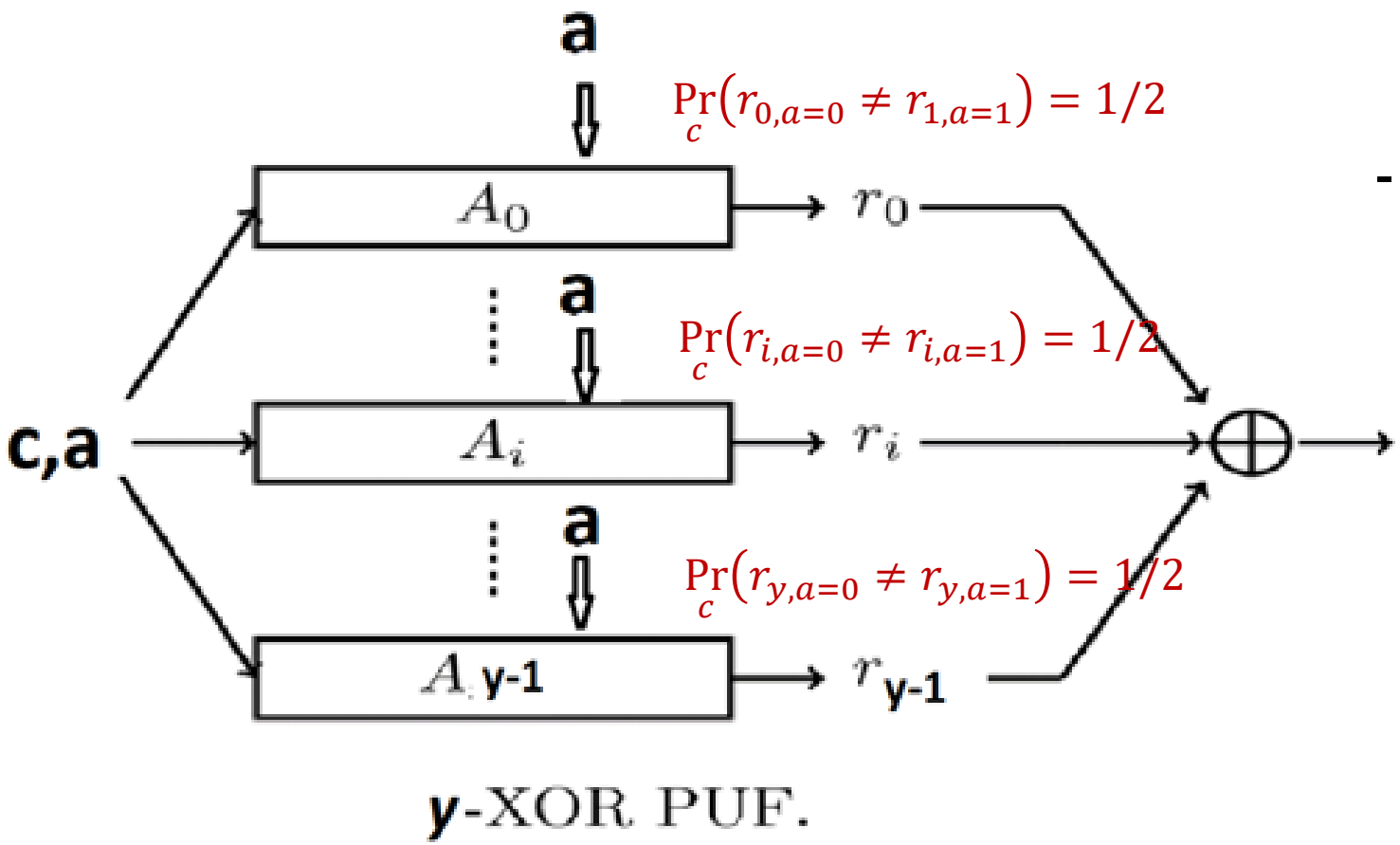
XOR APUF and iPUF



y-XOR PUF.

- r flips when odd # flipped r_i when a flips
- r not flip when even # flipped r_i when a flips

XOR APUF and iPUF

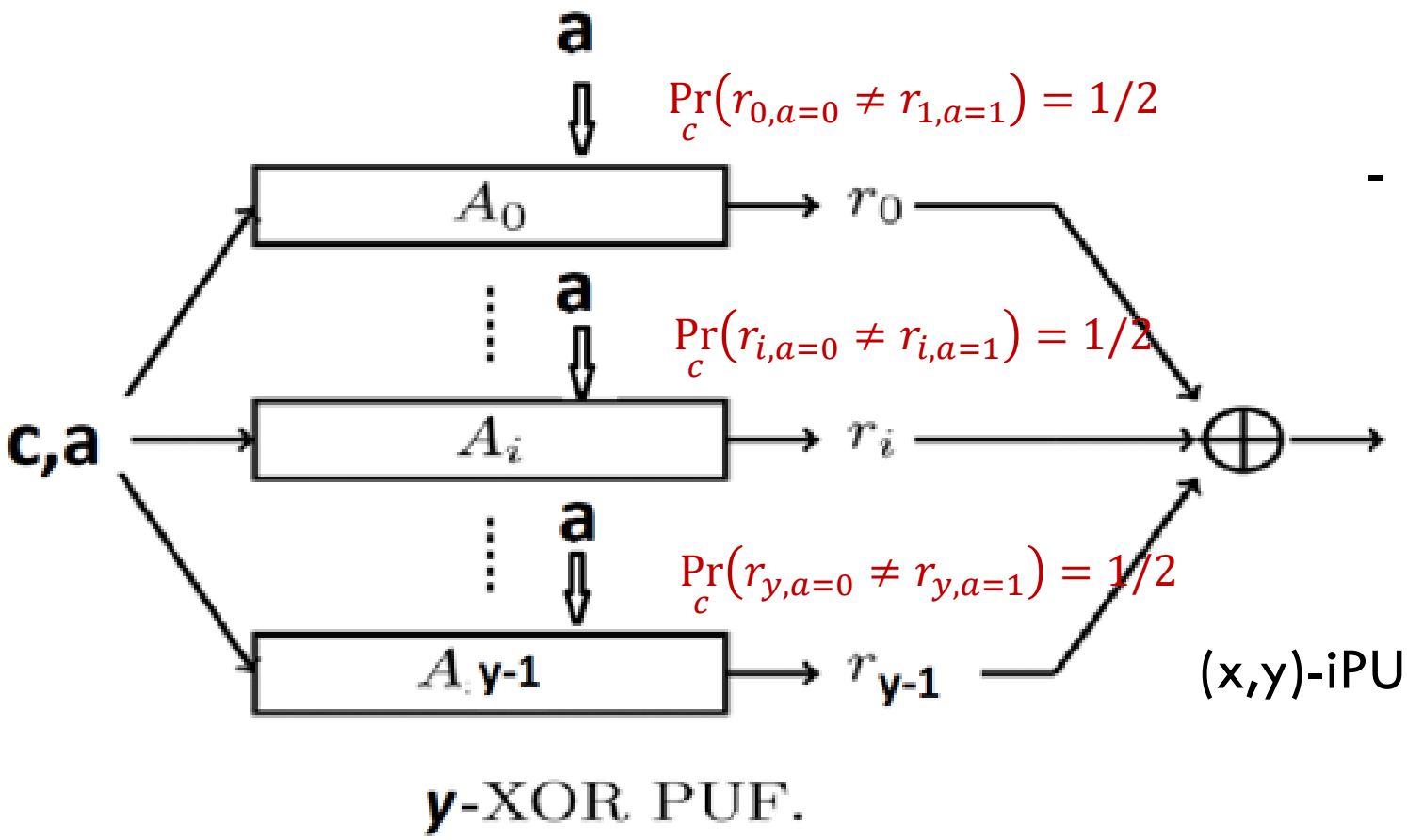


y-XOR PUF.

- r flips when odd # flipped r_i when a flips
- r not flip when even # flipped r_i when a flips

$$\Pr_c(r_{a=0} \neq r_{a=1}) = 1/2$$

XOR APUF and iPUF



y-XOR PUF.

- r flips when odd # flipped r_i when a flips
- r not flip when even # flipped r_i when a flips

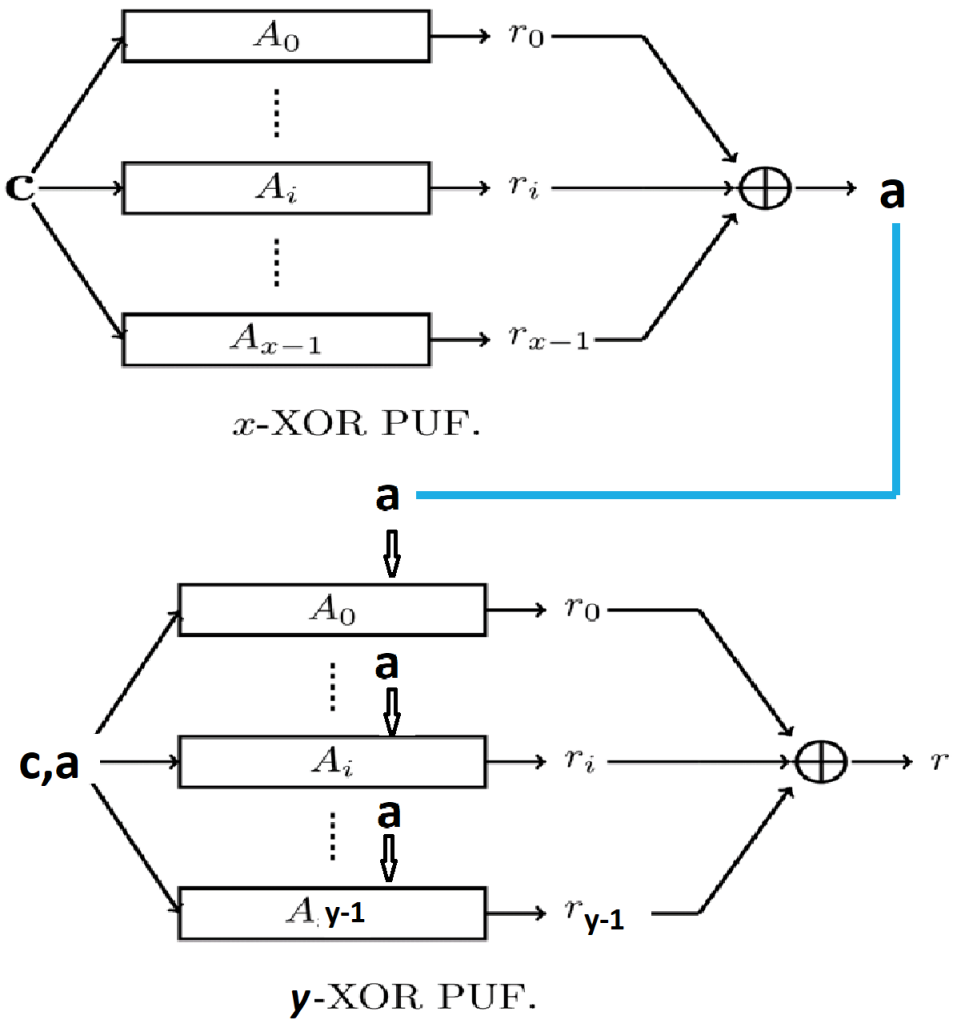
$\Pr_c(r_{a=0} \neq r_{a=1}) = 1/2$

(x,y)-iPUF = ½ y APUFs + ½ (x+y) APUFs
= (x/2+y) APUFs
= (x/2+y)-XOR APUFs

UCONN

XOR APUF and iPUF

(x,y)-iPUF



- All APUFs' outputs in x-XOR APUF contribute to a
- All APUF's outputs in y-XOR APUF contribute to r
- For a given challenge c,
  If flipping a makes iPUF's output r not flip due to SAC of APUF,
  then (x,y)-iPUF = y-XOR APUF
  because only y APUFs at y-XOR APUF contribute to r.

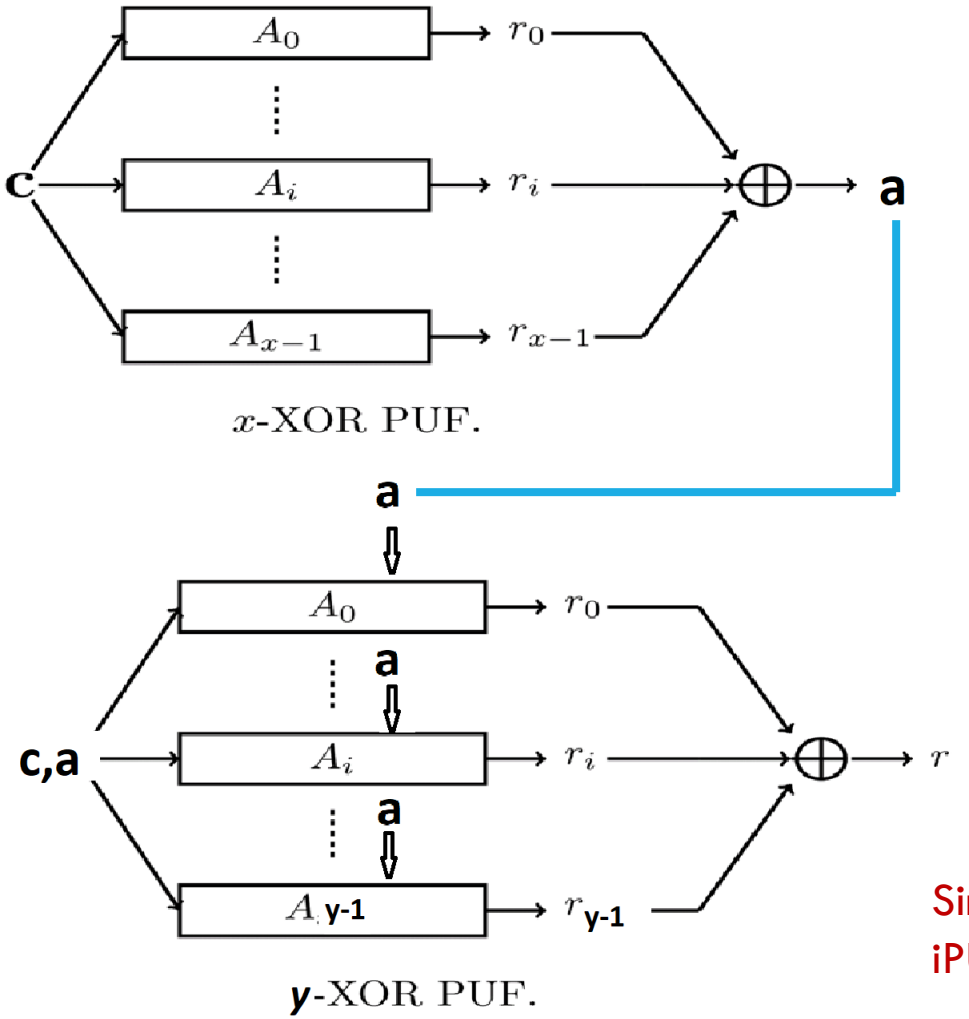
- For a given challenge c,
  If flipping a makes iPUF's output r flip due to SAC of APUF,
  then (x,y)-iPUF = (x+y)-XOR APUF
  because all x+y APUFs at x and y-XOR APUFs contribute to r

- If a is interposed at the middle (=n/2),
  then (x,y)-iPUF = (x/2+y)-XOR APUF

UCONN

XOR APUF and iPUF

(x,y)-iPUF



- All APUFs' outputs in x-XOR APUF contribute to a
- All APUF's outputs in y-XOR APUF contribute to r
- For a given challenge c,
  If flipping a makes iPUF's output r not flip due to SAC of APUF,
  then (x,y)-iPUF = y-XOR APUF
  because only y APUFs at y-XOR APUF contribute to r.

- For a given challenge c,
  If flipping a makes iPUF's output r flip due to SAC of APUF,
  then (x,y)-iPUF = (x+y)-XOR APUF
  because all x+y APUFs at x and y-XOR APUFs contribute to r

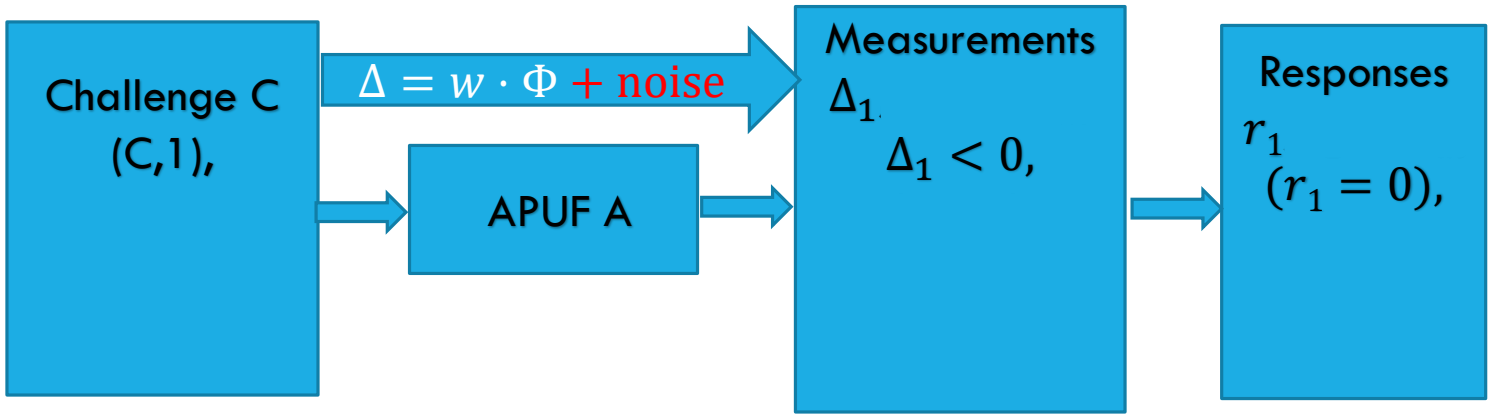
- If a is interposed at the middle (=n/2),
  then (x,y)-iPUF = (x/2+y)-XOR APUF

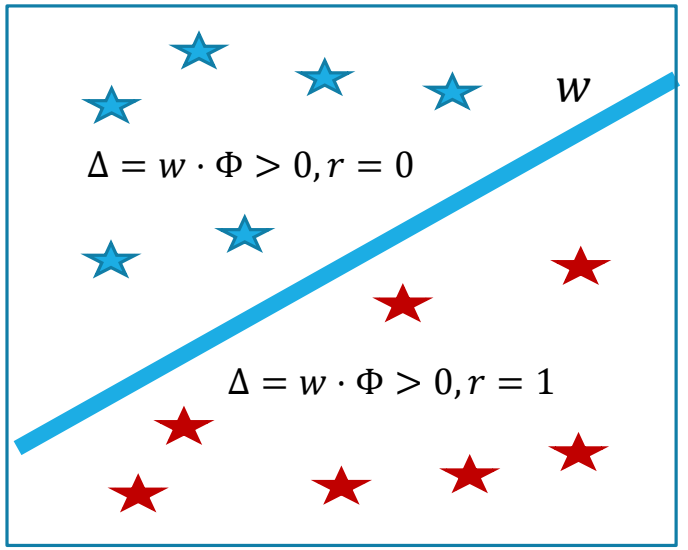
Since XOR APUF is secure against classical ML attack,
iPUF is also secure against classical ML attacks

UCONN

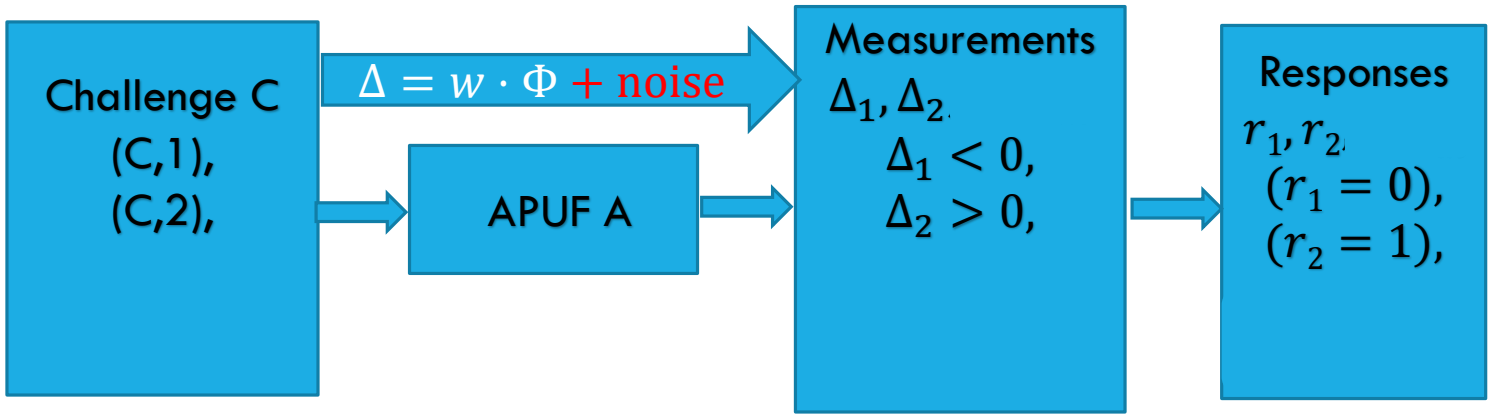
7. Short-term Reliability

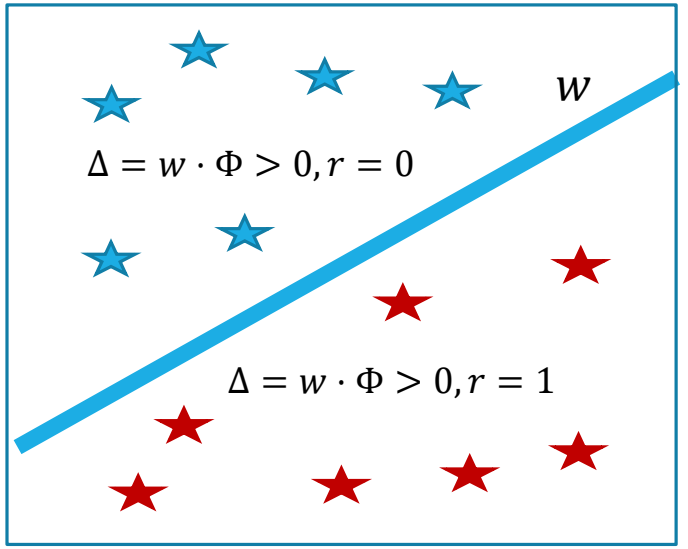
Arbiter: Repeatability – short-term Reliability



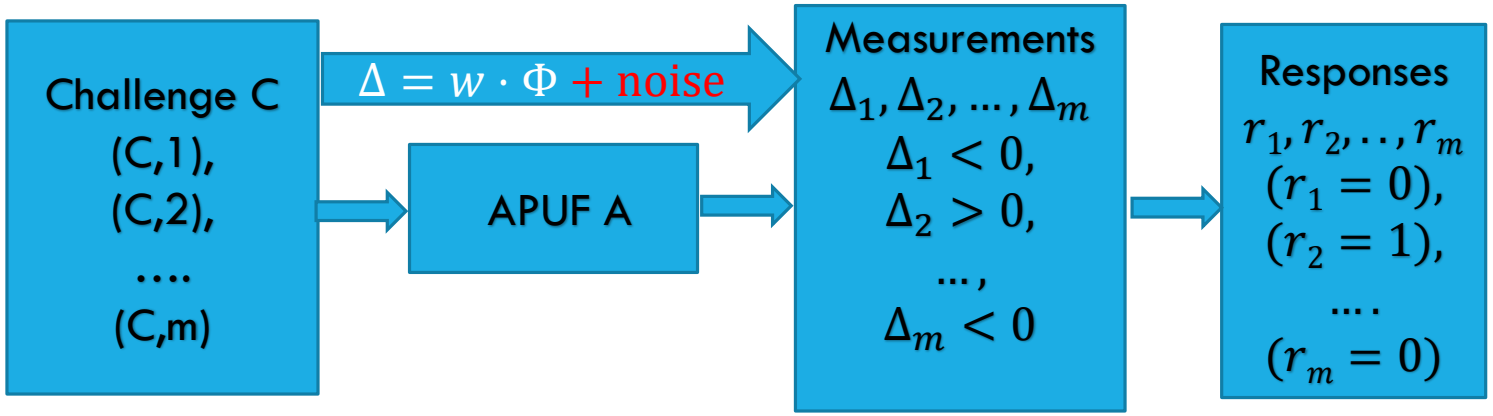


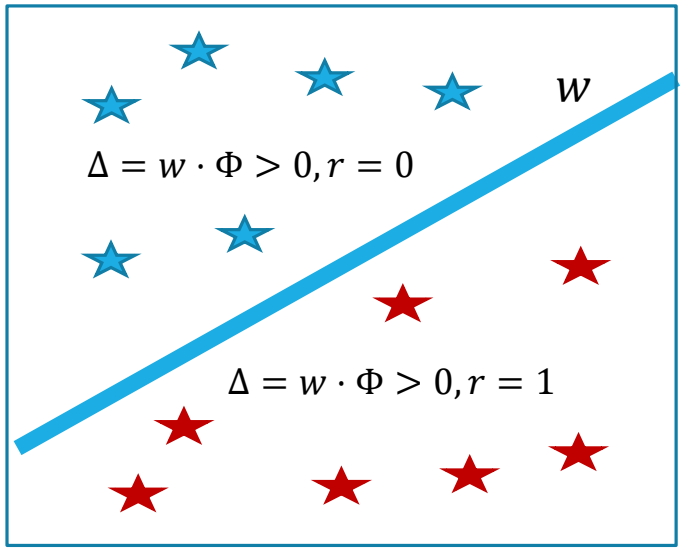
Arbiter: Repeatability – short-term Reliability



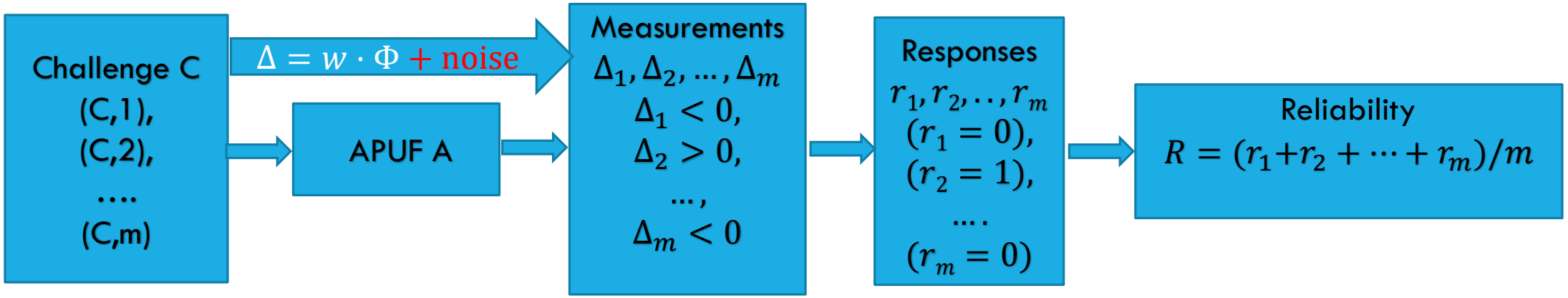


Arbiter: Repeatability – short-term Reliability

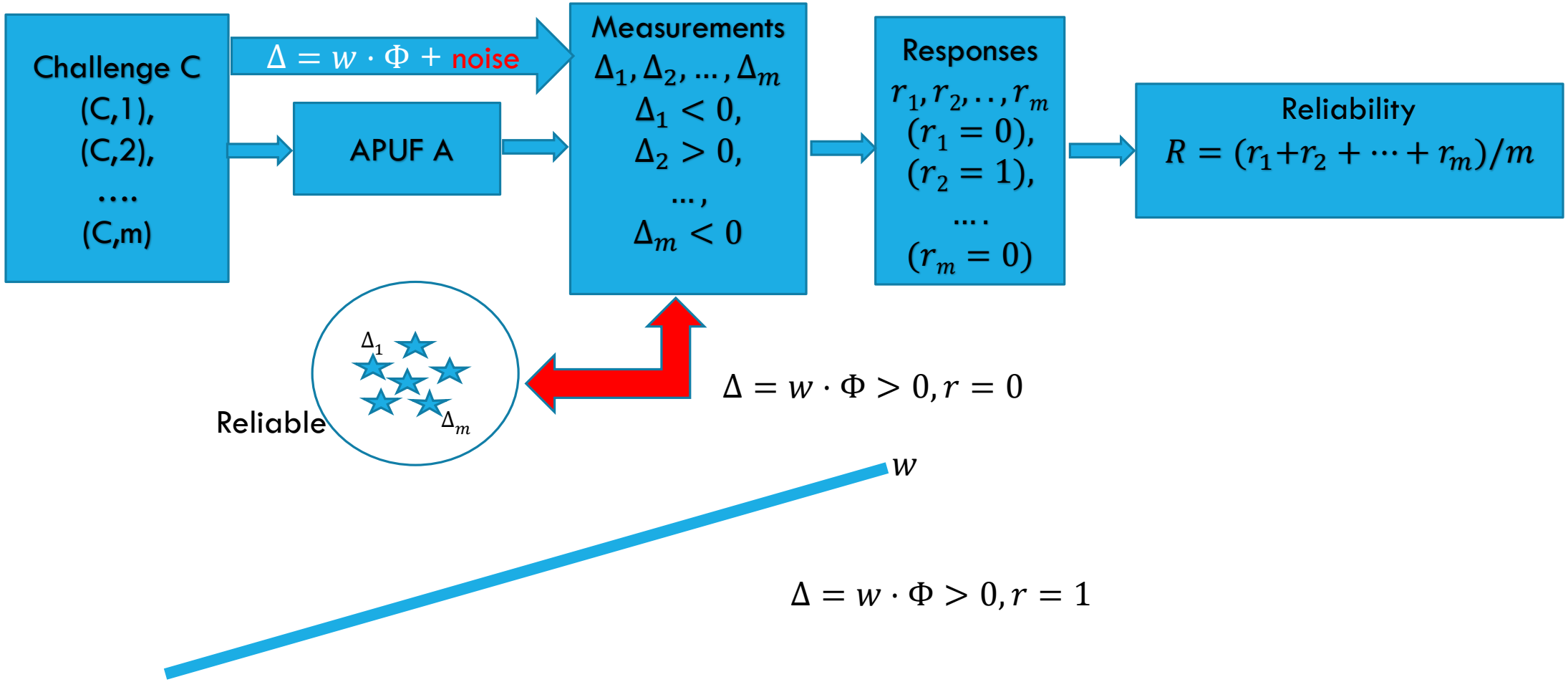




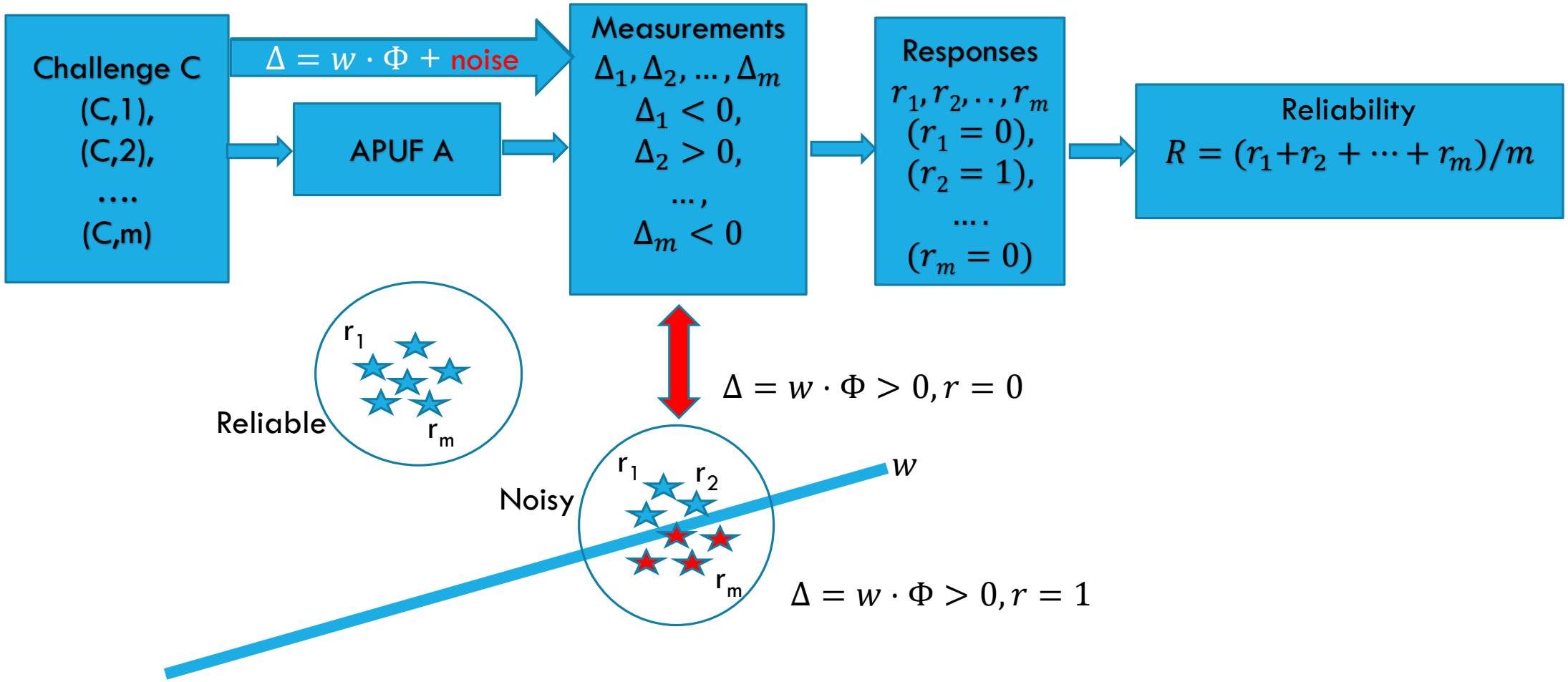
Arbiter: Repeatability – short-term Reliability



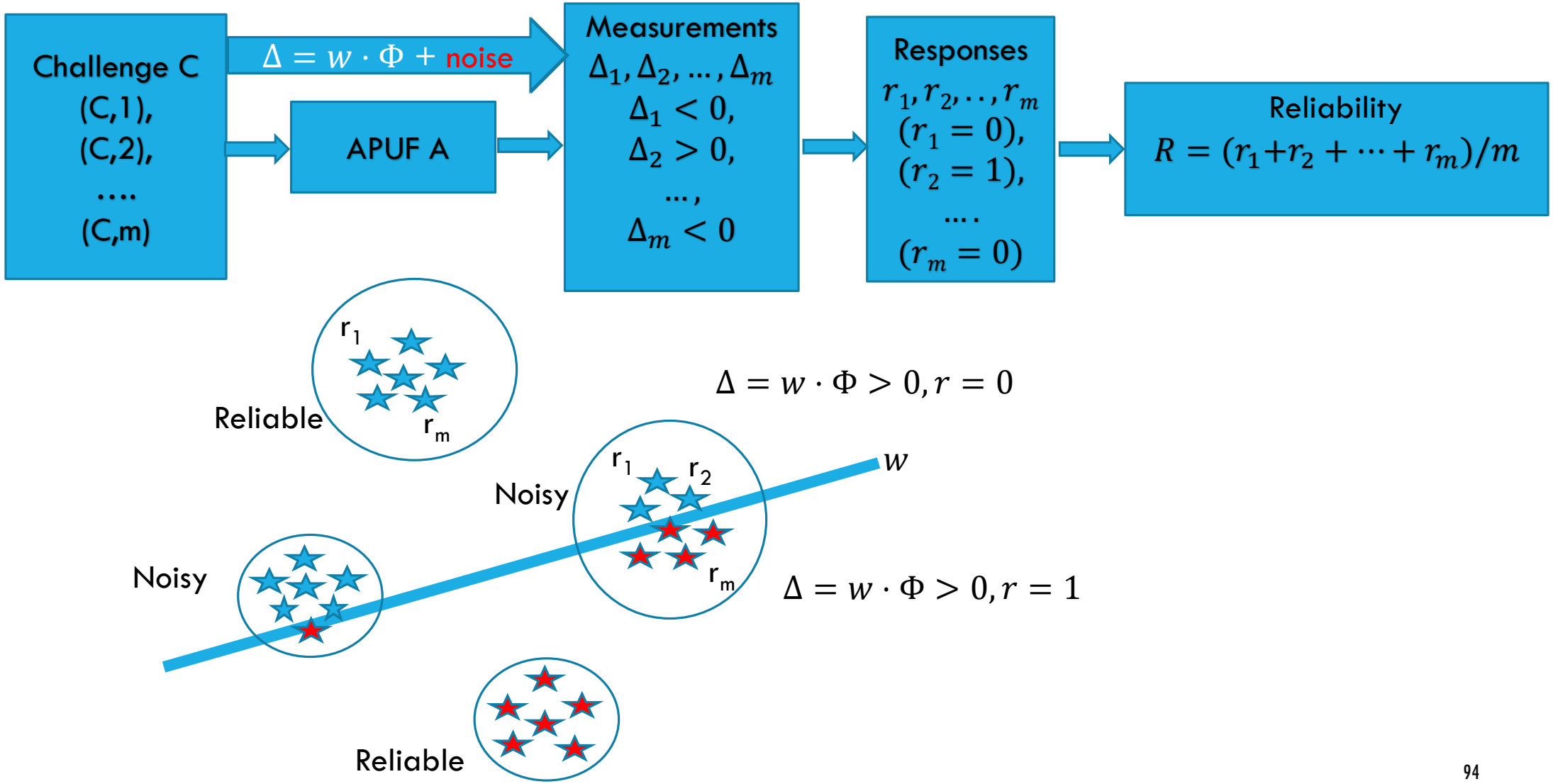
Arbiter: Repeatability – short-term Reliability



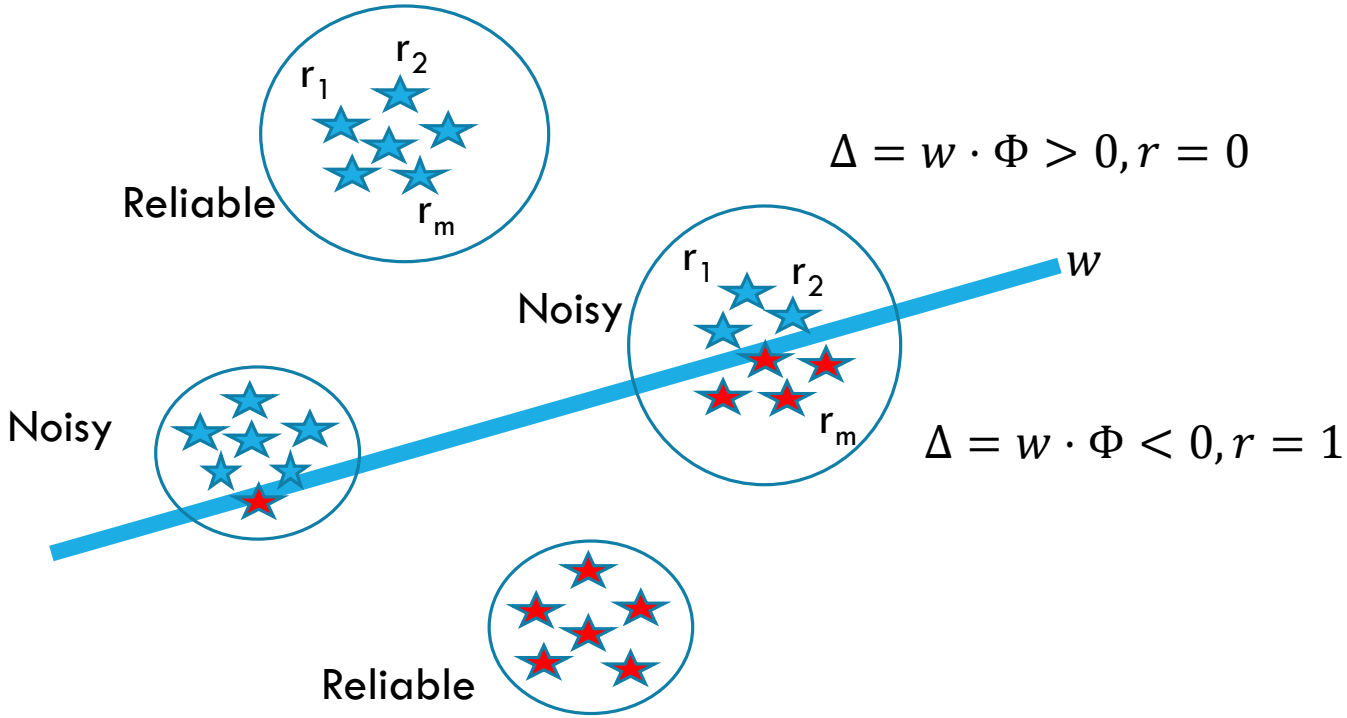
Arbiter: Repeatability – short-term Reliability



Arbiter: Repeatability – short-term Reliability



Arbiter: Repeatability – short-term Reliability



Reliability of C and w are related

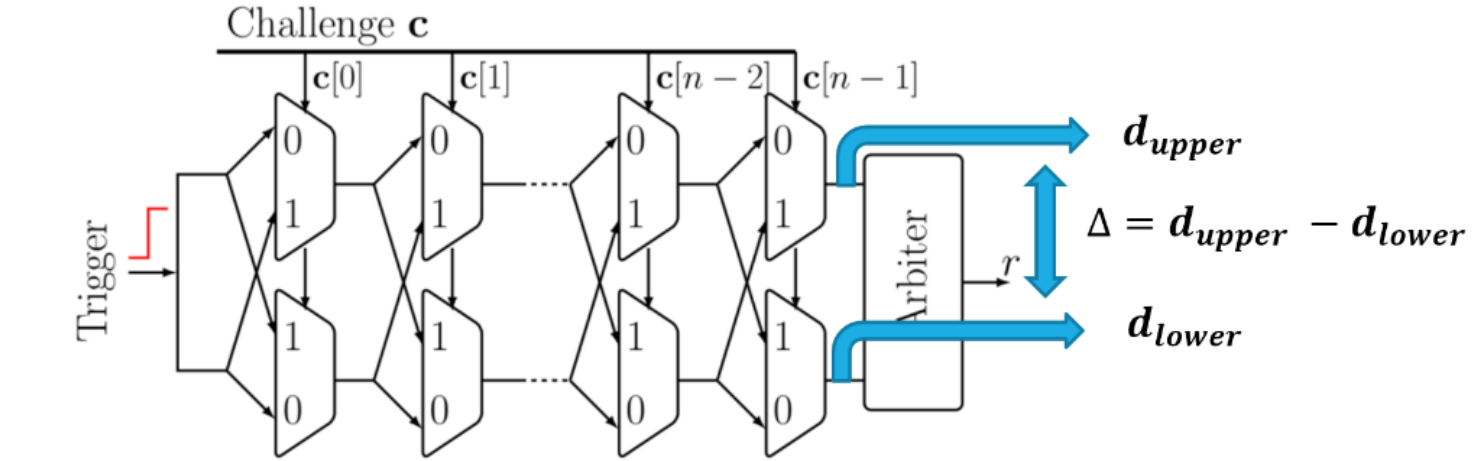
CMA-ES + many pairs of (c,R): Reliability based modeling attack
The Gap Between Promise and Reality: On the Insecurity of XOR Arbiter PUFs CHES, 2015, Georg T. Becker

UCONN

8. Reliability based Modeling Attacks

UCONN

APUF



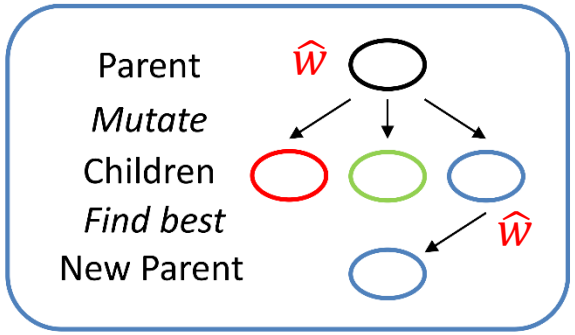
A n-stage classic Arbiter PUF with challenge $\mathbf{c} \in \{0,1\}^n$.

- $\Delta > 0 \rightarrow r = 1. Otherwise\ r = 0$
- $\Delta = \boldsymbol{d_{upper}} - \boldsymbol{d_{lower}} = \boldsymbol{w} \cdot \boldsymbol{\Phi}$

Covariance Matrix Adaptation Evolution Strategy (CMA-ES) Algorithm

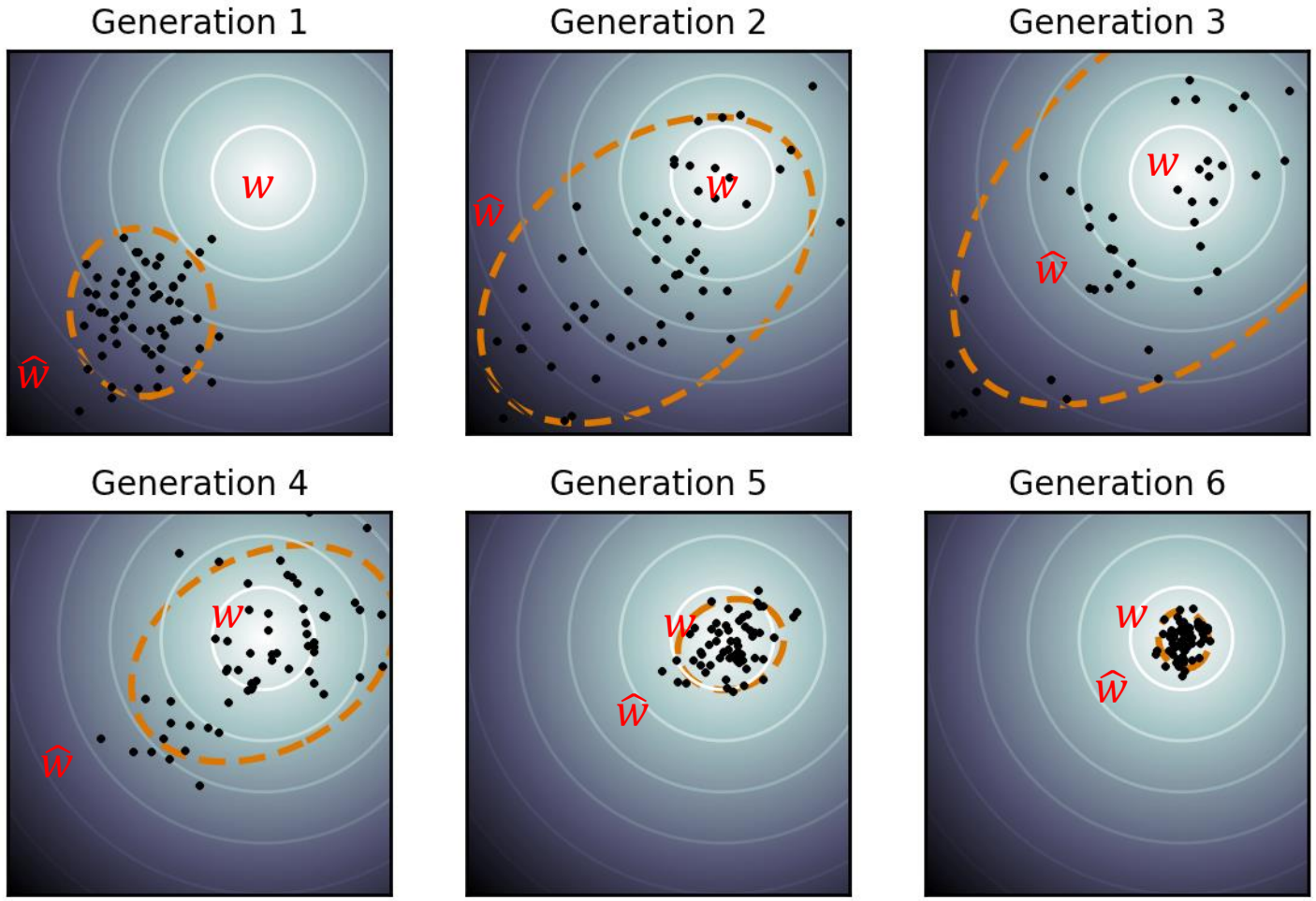
w: $target$

$\hat{w}$: $estimator\ or\ model$

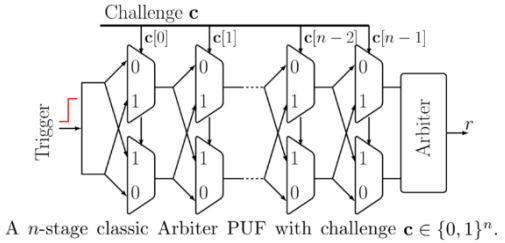


[4]

$\hat{w} = w$



Reliability-based modeling attack on APUFs using CMAES

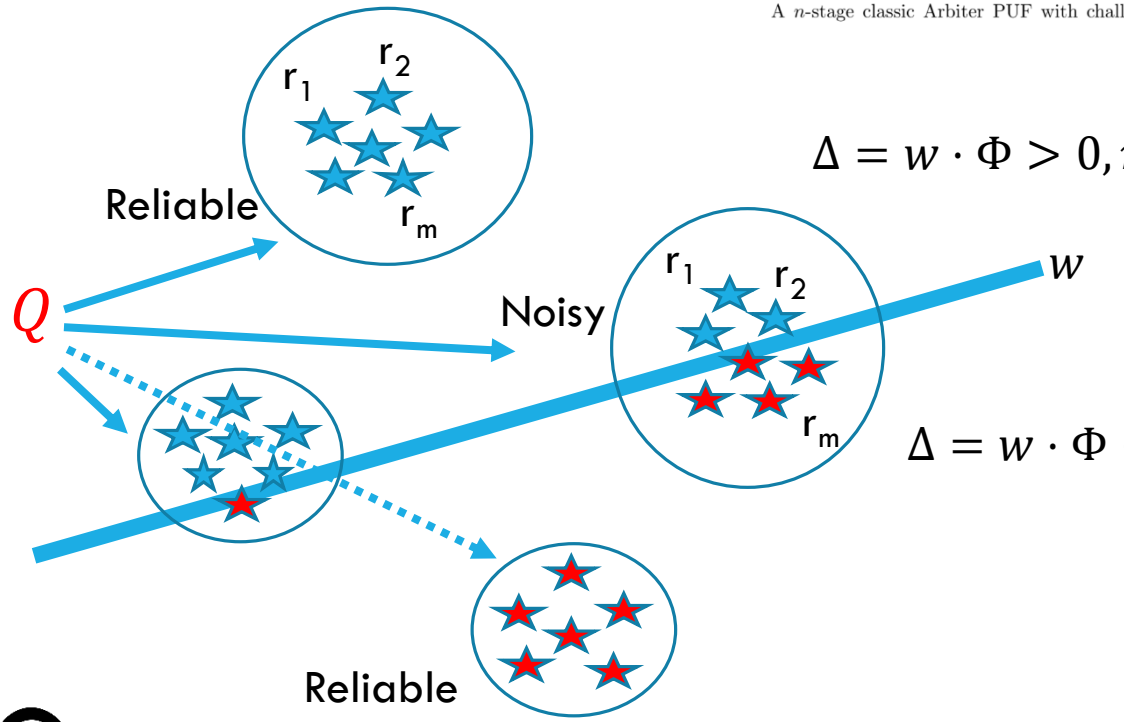


A n-stage classic Arbiter PUF with challenge $\mathbf{c} \in \{0,1\}^n$.

Q: $set\ of\ CRPs$,
w: $APUF$

$$\Delta = w \cdot \Phi > 0, r = 0$$

$$\Delta = w \cdot \Phi < 0, r = 1$$



Q, w

Q_n : noisy challenges

Q_r : reliable challenges

Reliability-based modeling attack on APUFs using CMAES

Iteration 1

Target

Q, w

Q_n : noisy

Q_r : reliable

Model

$Q, \epsilon_1, \hat{w}_1$

$Q \rightarrow challenge\ c \rightarrow \Phi(c) \rightarrow \Delta = \hat{w}_1 \cdot \Phi(c)$

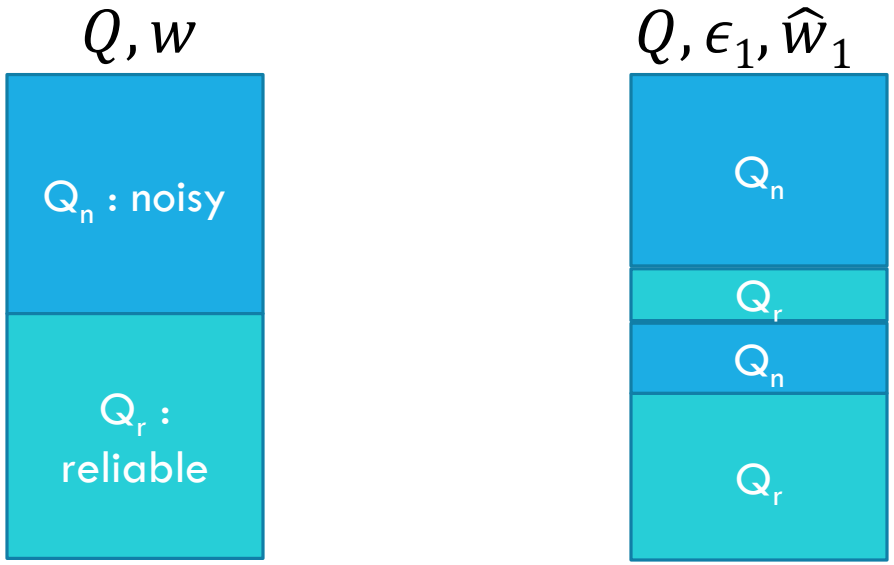
$|\Delta| \leq \epsilon_1 \rightarrow$ $challenge\ c\ is\ noisy$ → Q_n : noisy

$|\Delta| > \epsilon_1 \rightarrow$ $challenge\ c\ is\ reliable$ → Q_r : reliable

UCONN

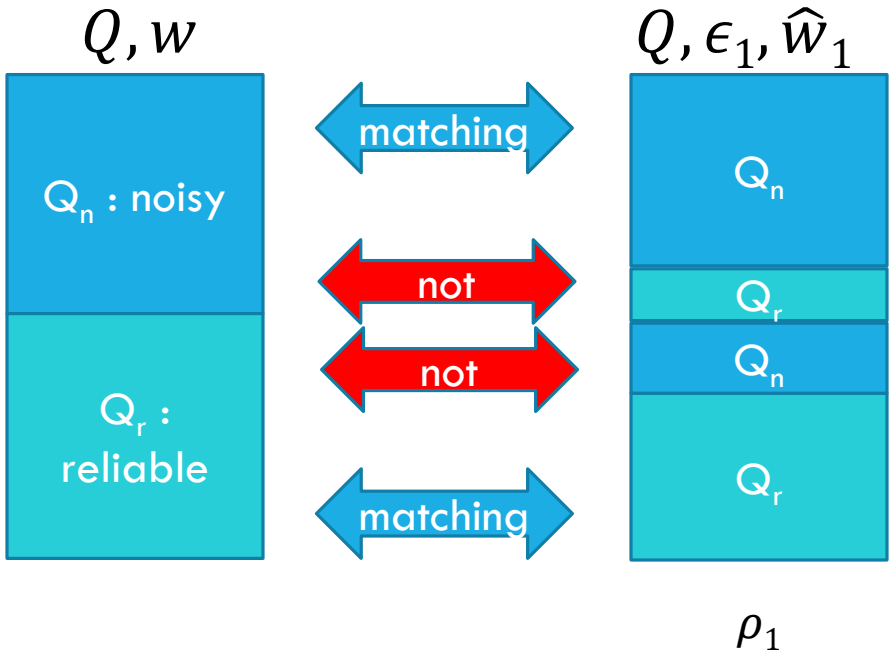
Reliability-based modeling attack on APUFs using CMAES

Iteration 1



Reliability-based modeling attack on APUFs using CMAES

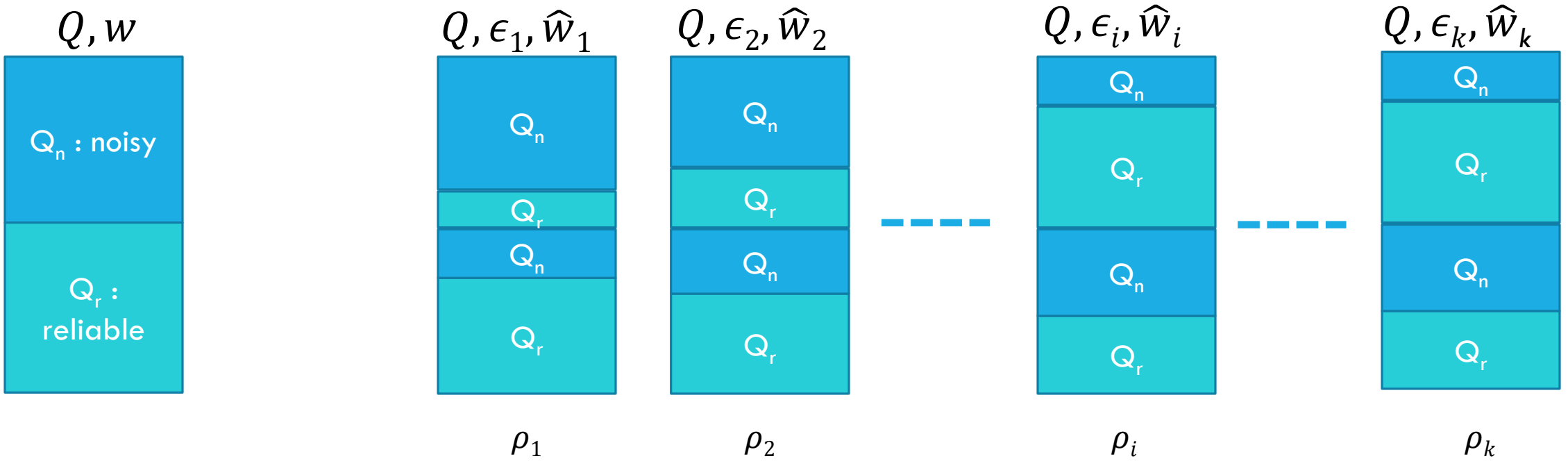
Iteration 1



$Compute\ the\ matching\ rate\ \rho_1\ between\ [Q, w]\ and\ [Q, \epsilon_1, \hat{w}_1]$

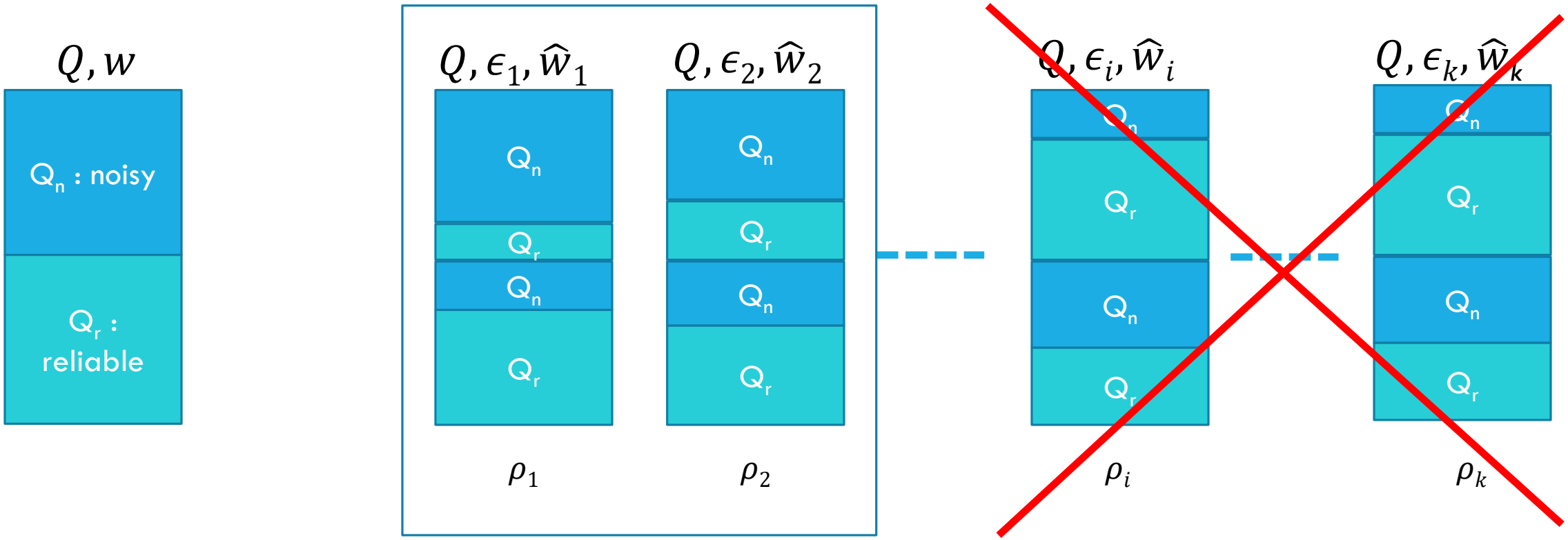
Reliability-based modeling attack on APUFs using CMAES

Iteration 1



Reliability-based modeling attack on APUFs using CMAES

Iteration 1



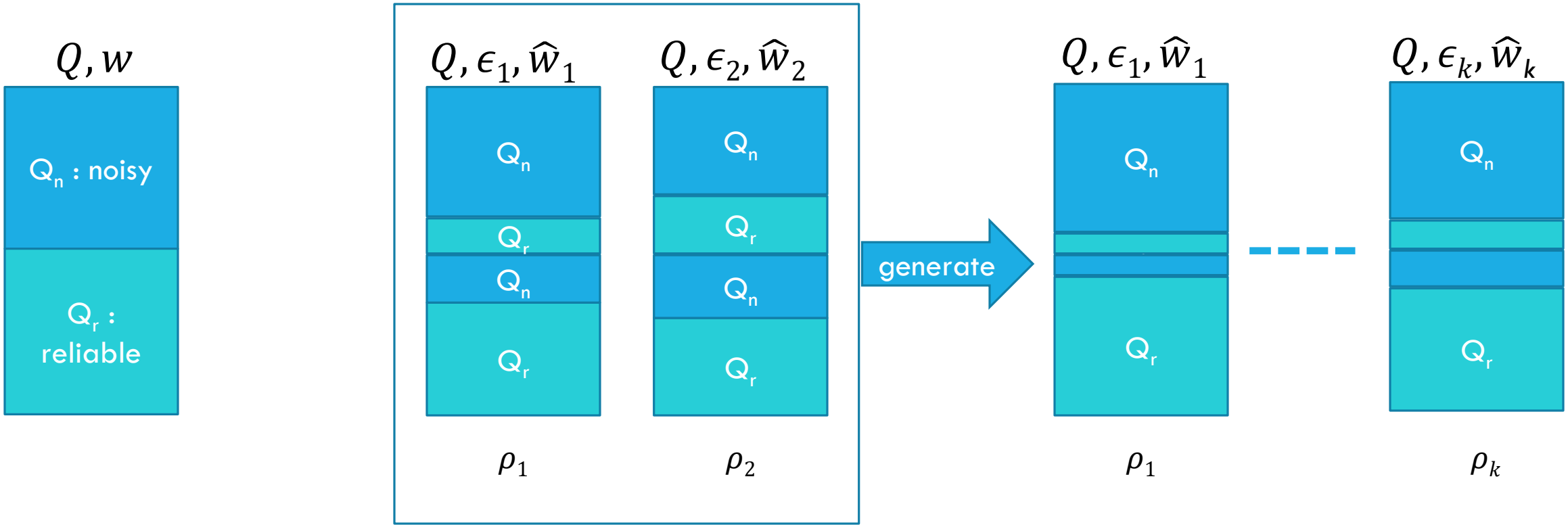
High matching rate: Kept

Low matching rate: Discarded

UCONN

Reliability-based modeling attack on APUFs using CMAES

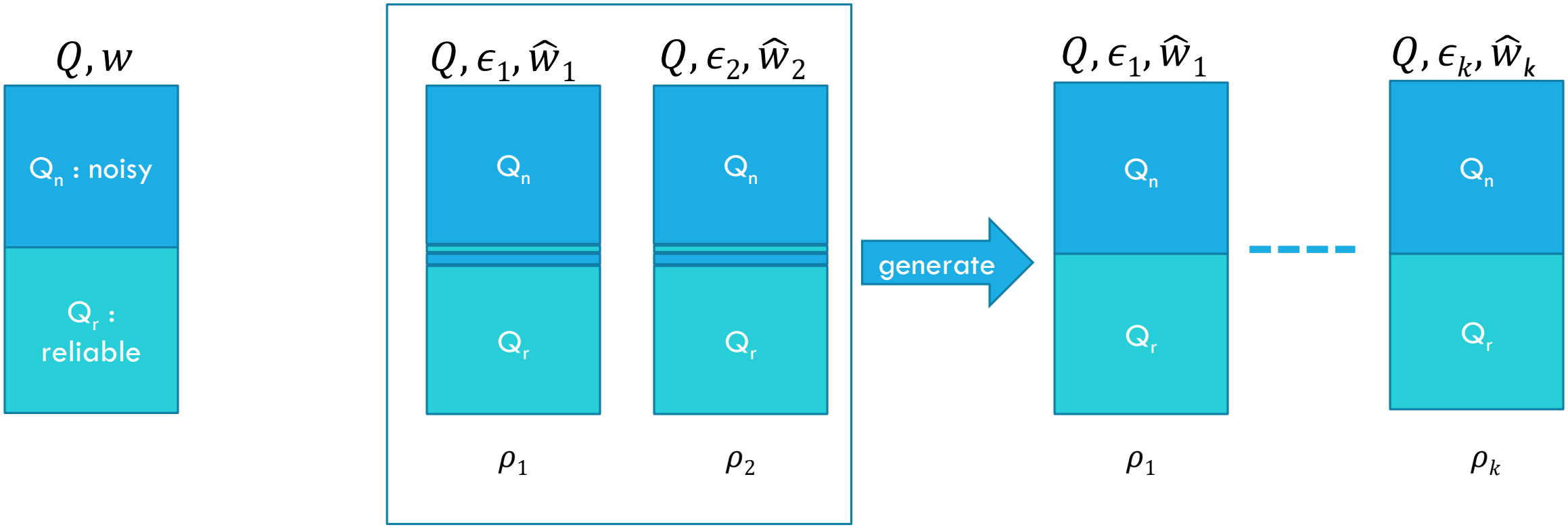
Iteration 2



High matching rate: Kept

Reliability-based modeling attack on APUFs using CMAES

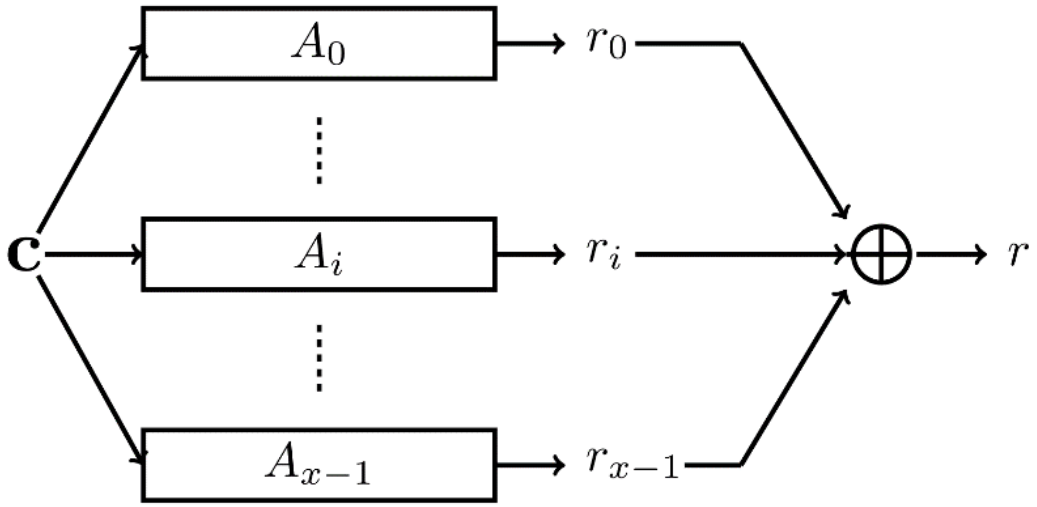
Iteration M



High matching rate: Kept

UCONN

x-XOR APUF



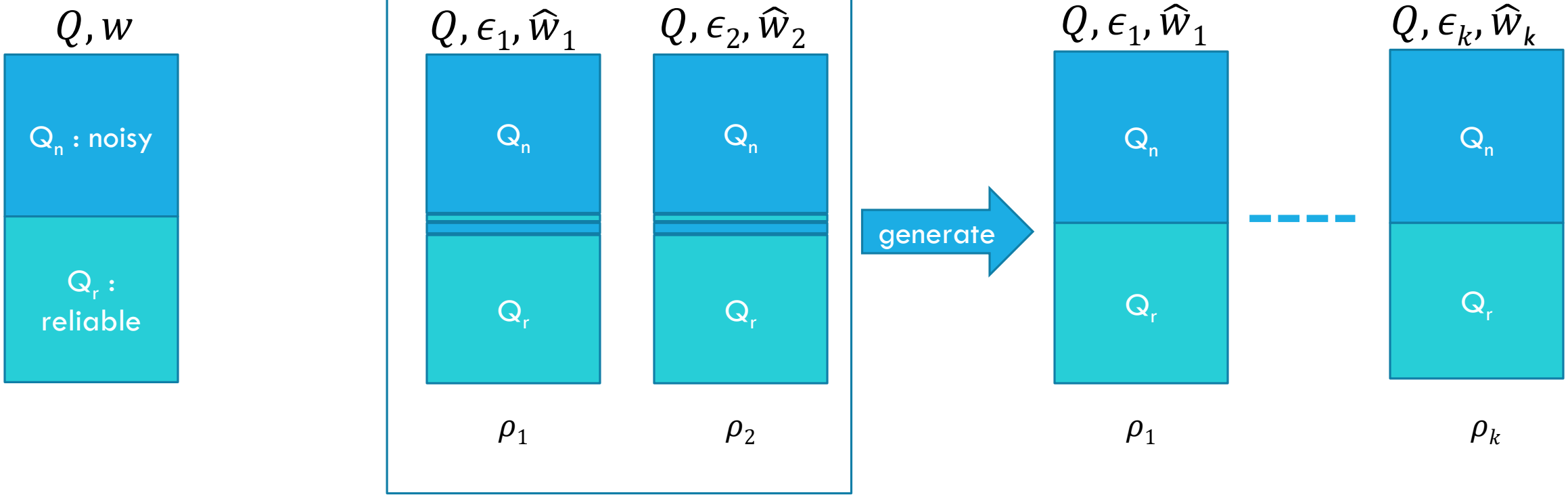
x-XOR PUF.

Reliability-based modeling attack on XOR APUFs using CMAES

Iteration i

APUF → Q

$\widehat{w}_1, \ldots, \widehat{w}_k$ are model of APUF

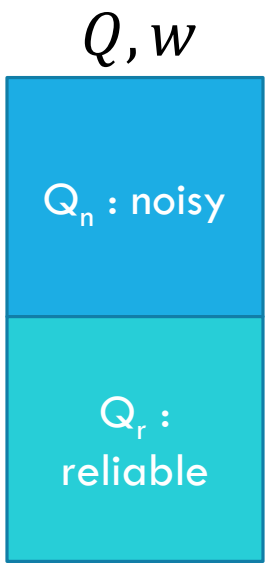


High matching rate: Kept

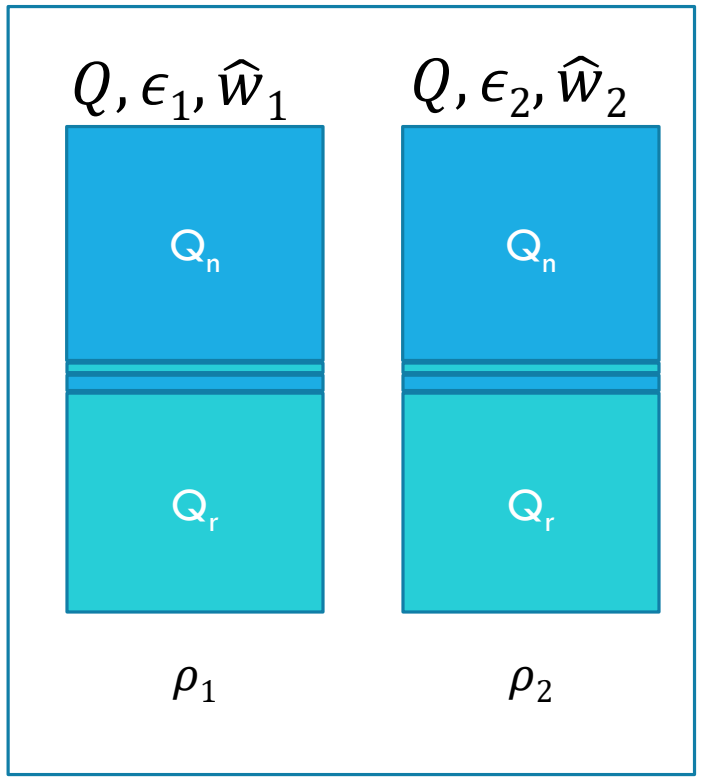
CMA-ES attack on APUF

Reliability-based modeling attack on XOR APUFs using CMAES

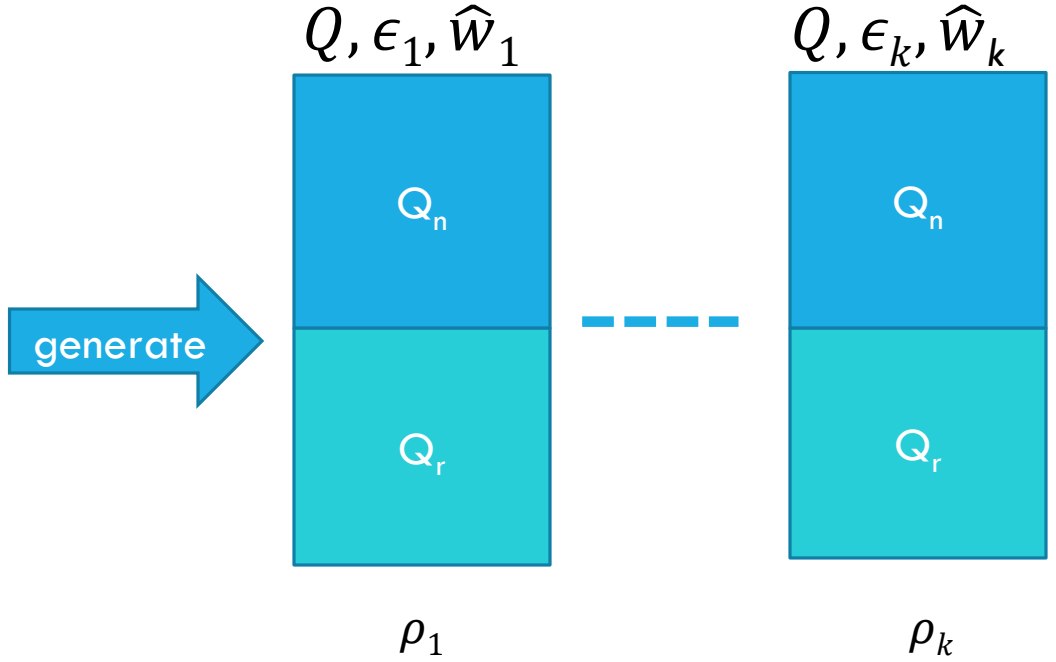
XOR APUF → Q



Iteration i



$\hat{w}_1, \ldots, \hat{w}_k$ are still model of APUF

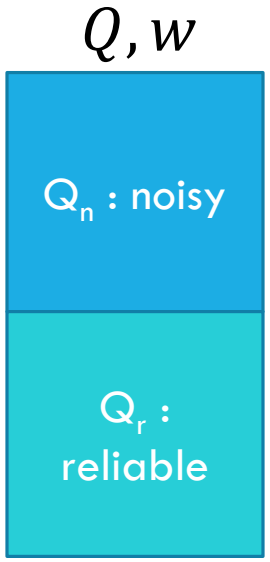


High matching rate: Kept

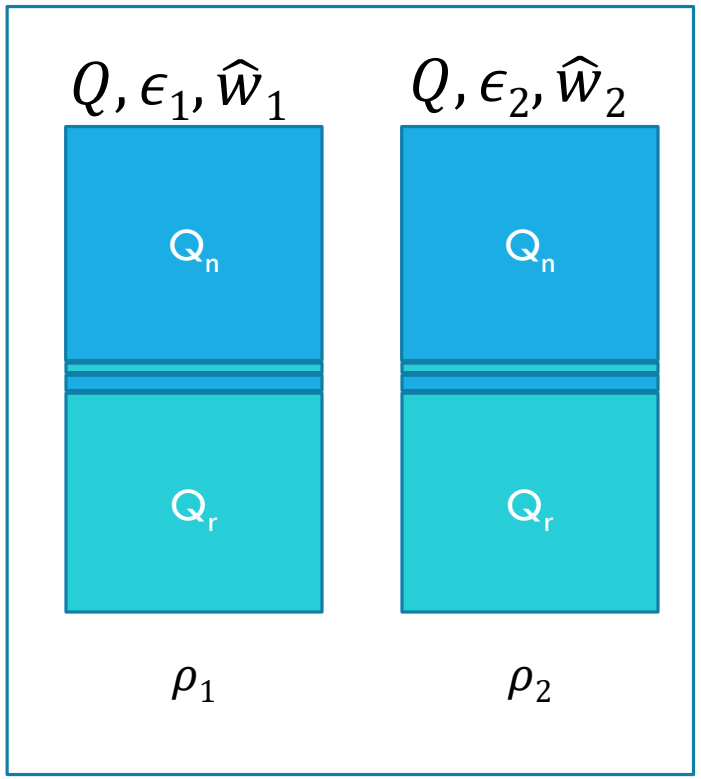
CMA-ES attack on XOR APUF

Reliability-based modeling attack on XOR APUFs using CMAES

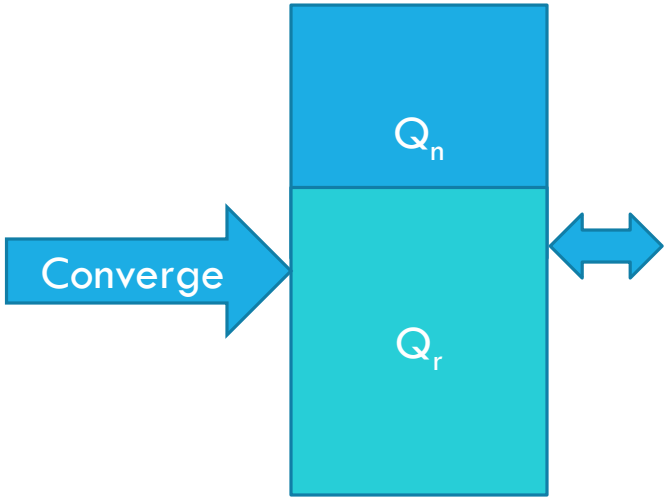
XOR APUF $\rightarrow$ Q



Iteration M



$\widehat{w}_1, \ldots, \widehat{w}_k$ are still model of APUF



One certain APUF i among x APUFs in x-XOR APUF

High matching rate: Kept

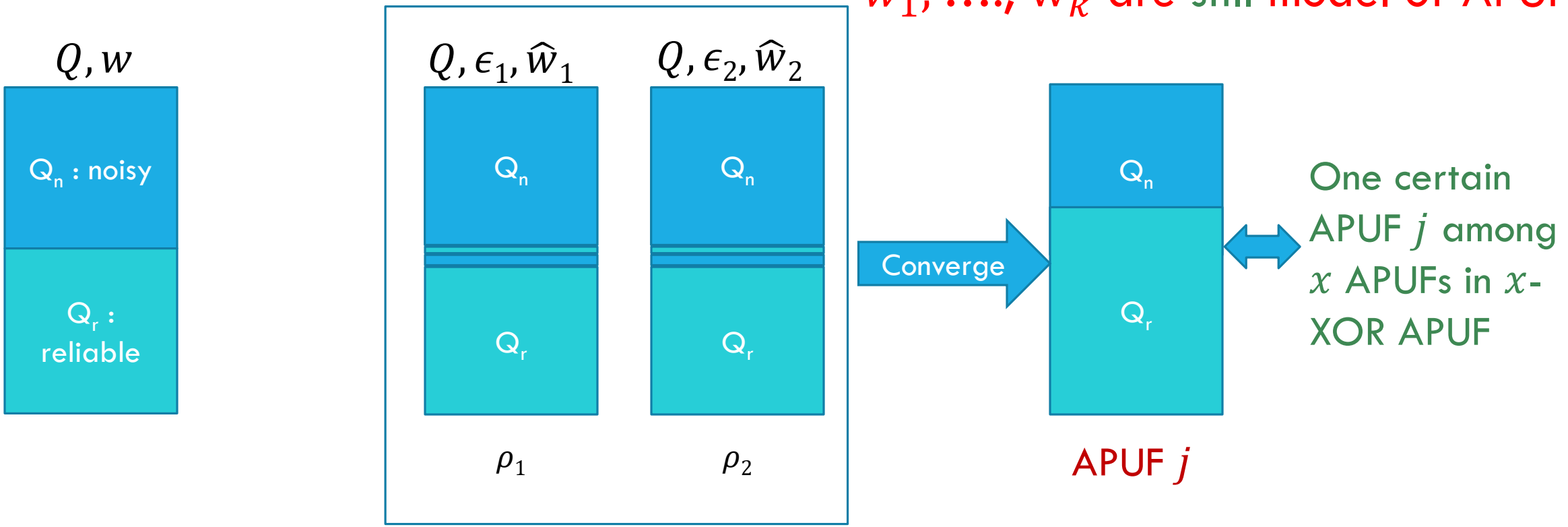
CMA-ES attack on XOR APUF

Reliability-based modeling attack on XOR APUFs using CMAES

XOR APUF → Q [Change]

Iteration M

$\widehat{w}_1, \ldots, \widehat{w}_k$ are still model of APUF



High matching rate: Kept

CMA-ES attack on XOR APUF

UCONN

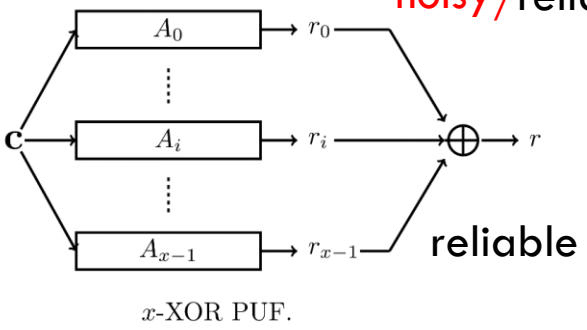
Understanding Reliability based modeling attack on XOR PUF

- Question 1: How does the attack on XOR PUF work?
- Question 2: How can we make the attack on XOR PUF fail?

UCONN

Question 1: How does the attack on XOR PUF work?

The noisy and reliable challenges in XOR PUF



x-XOR PUF.

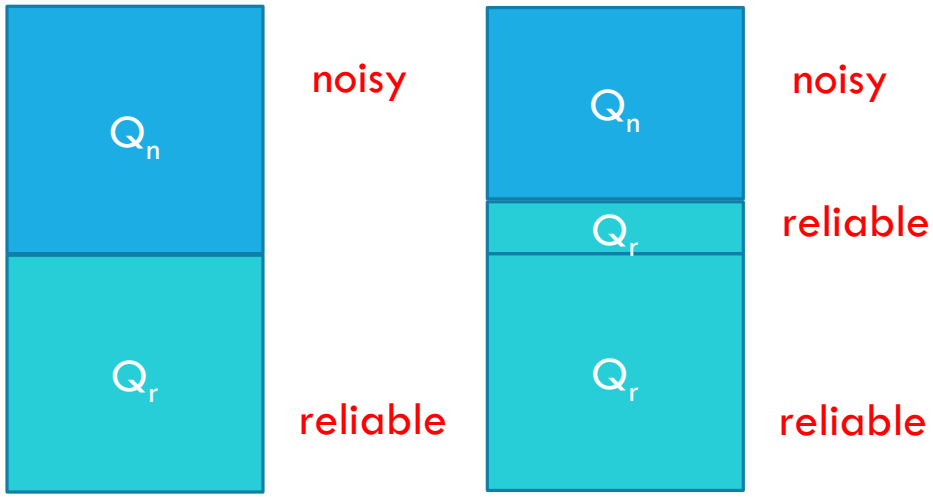
noisy /reliable

noisy /reliable

reliable

XOR APUF

APUF 1



noisy

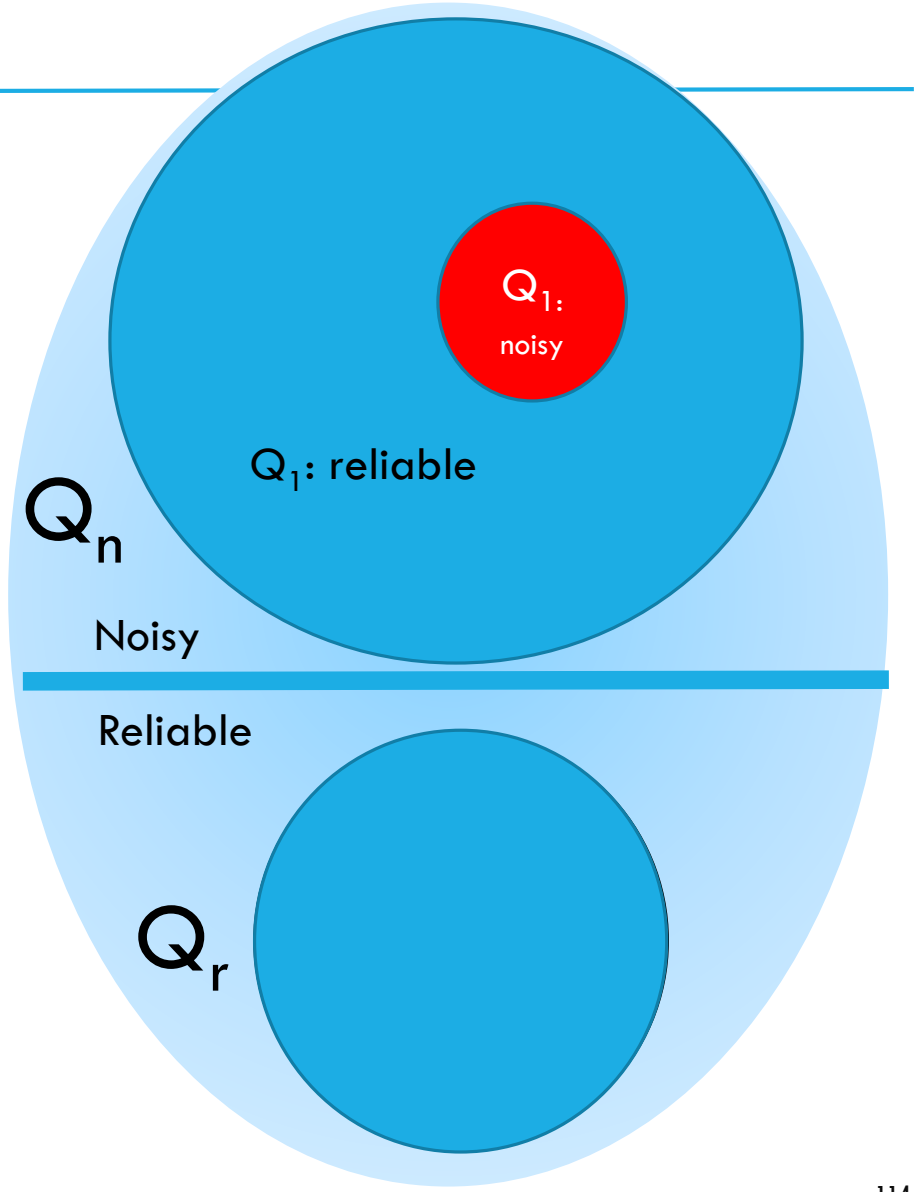
reliable

noisy

reliable

reliable

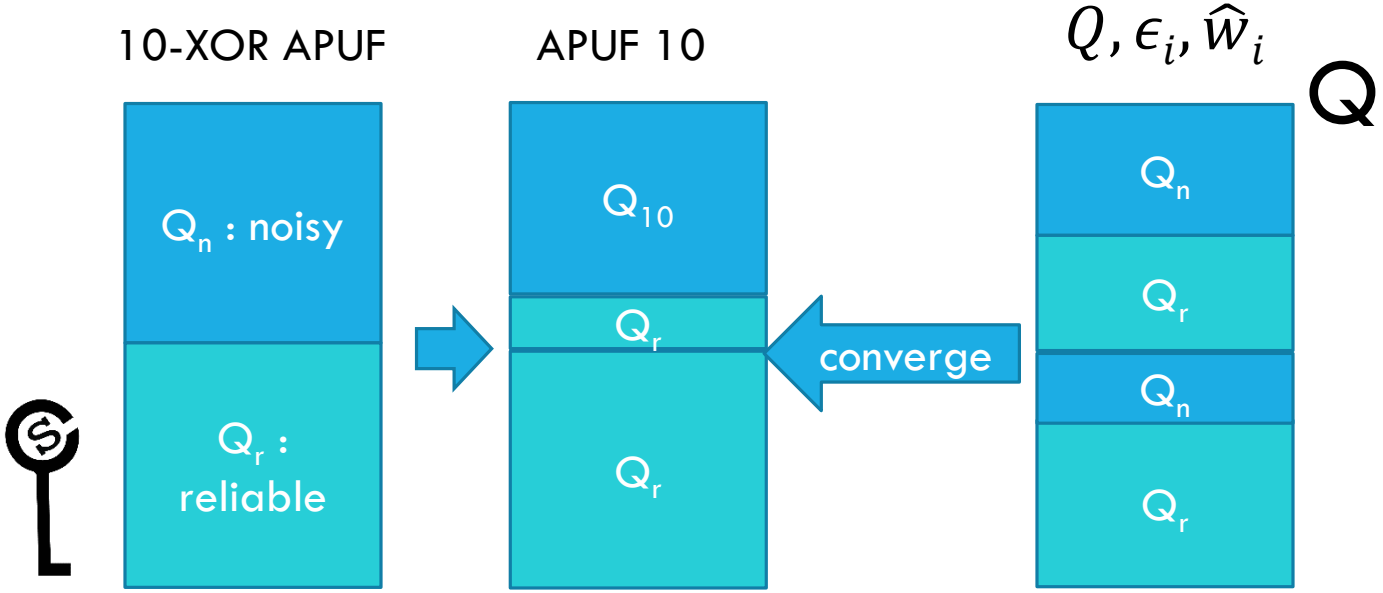
Q

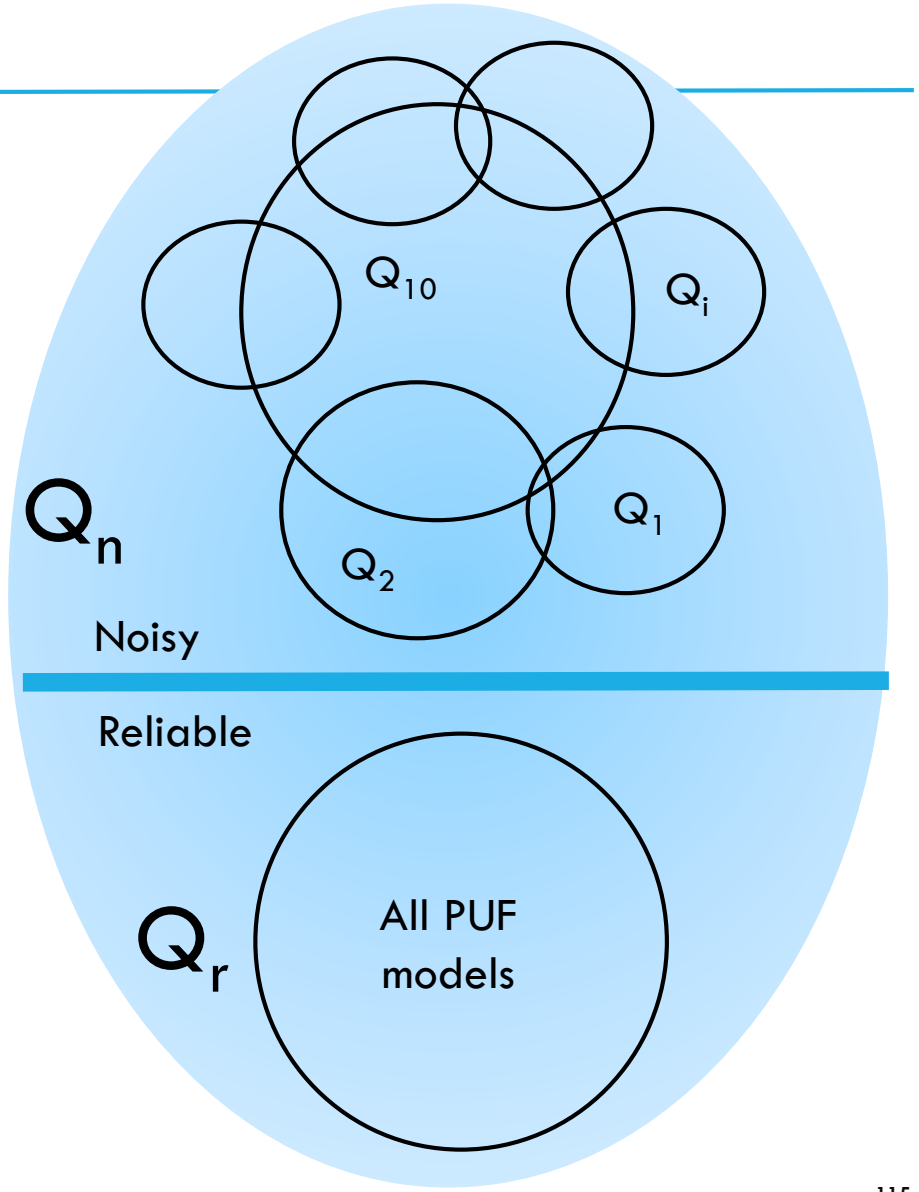


Challenge-Reliability Pairs Training Data

Key idea of the attack on XOR PUF

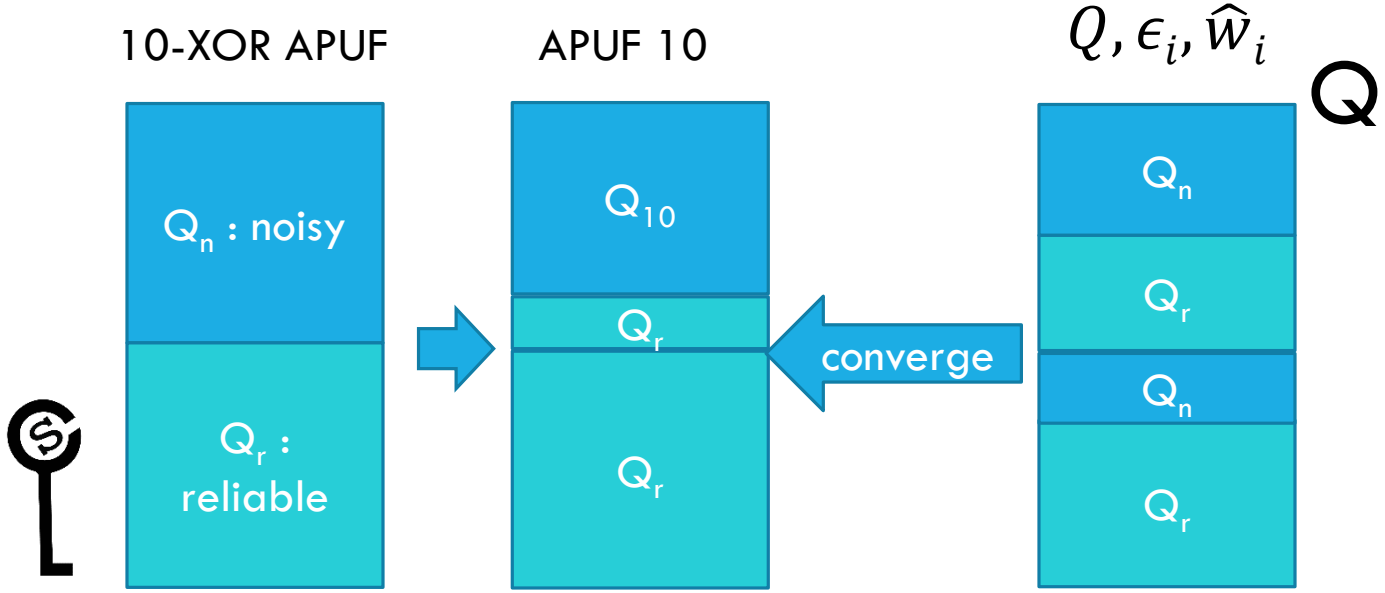
- (1) All the models $\widehat{w}_i$ in CMA-ES are models of APUF

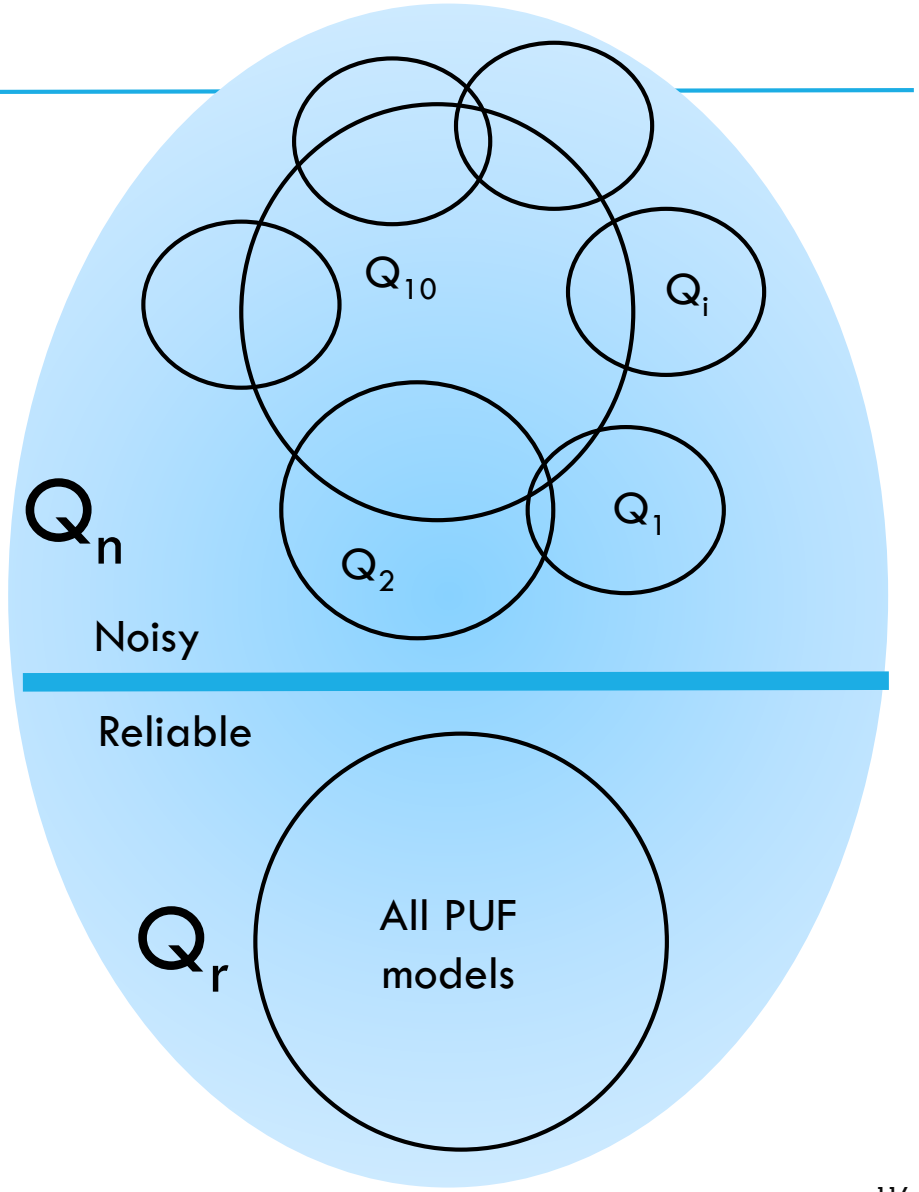




Key idea of the attack on XOR PUF

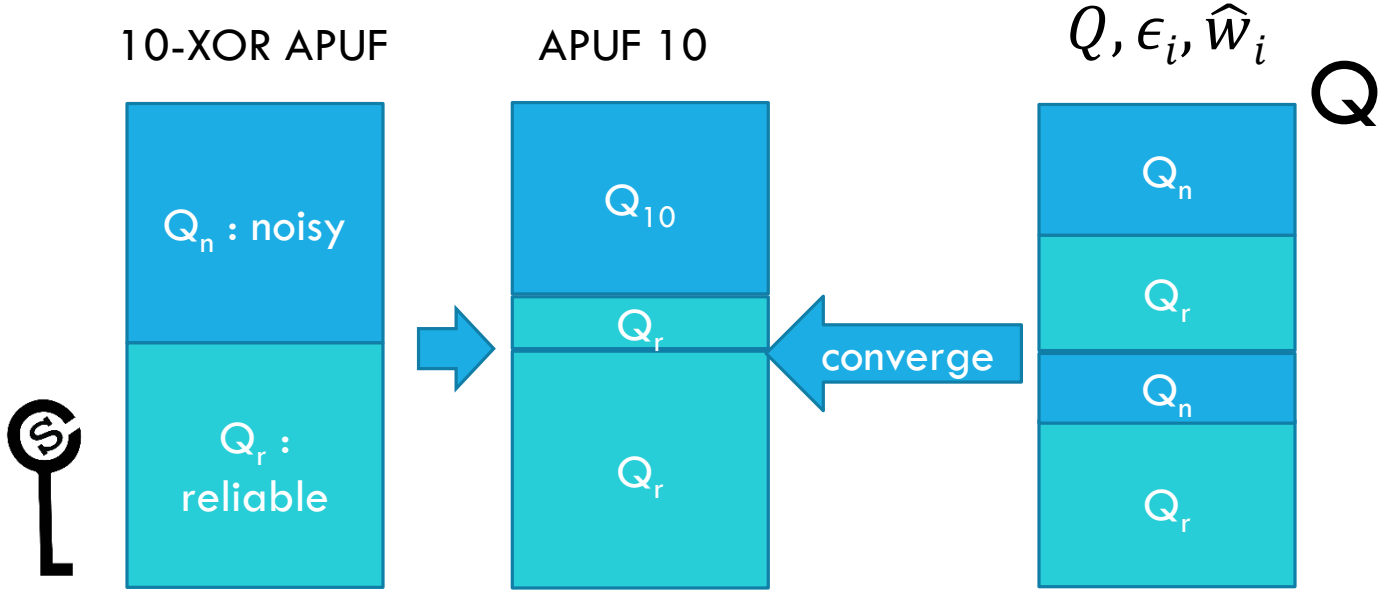
- (1) All the models $\widehat{w}_i$ in CMA-ES are models of APUF
- (2) $\widehat{w}_i$ can only converge to an APUF instance

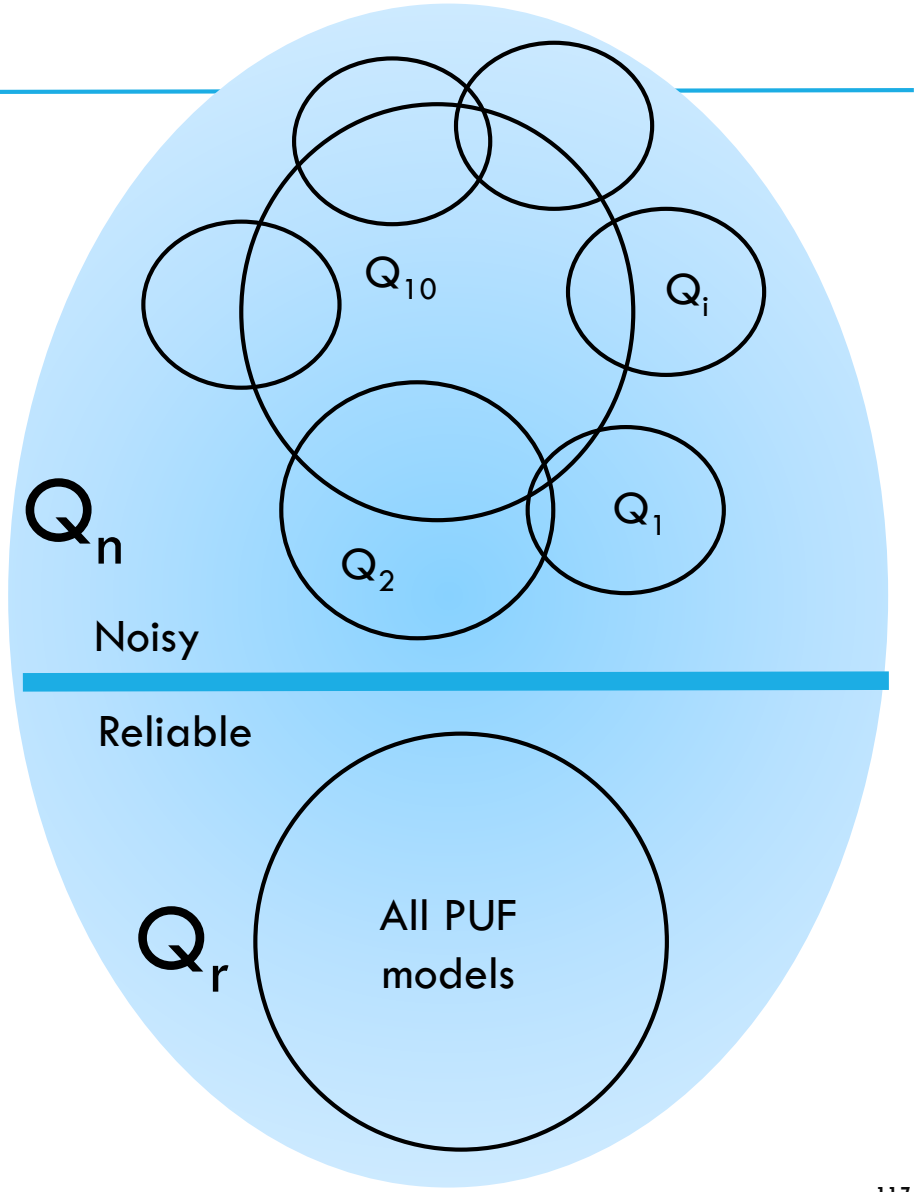




Key idea of the attack on XOR PUF

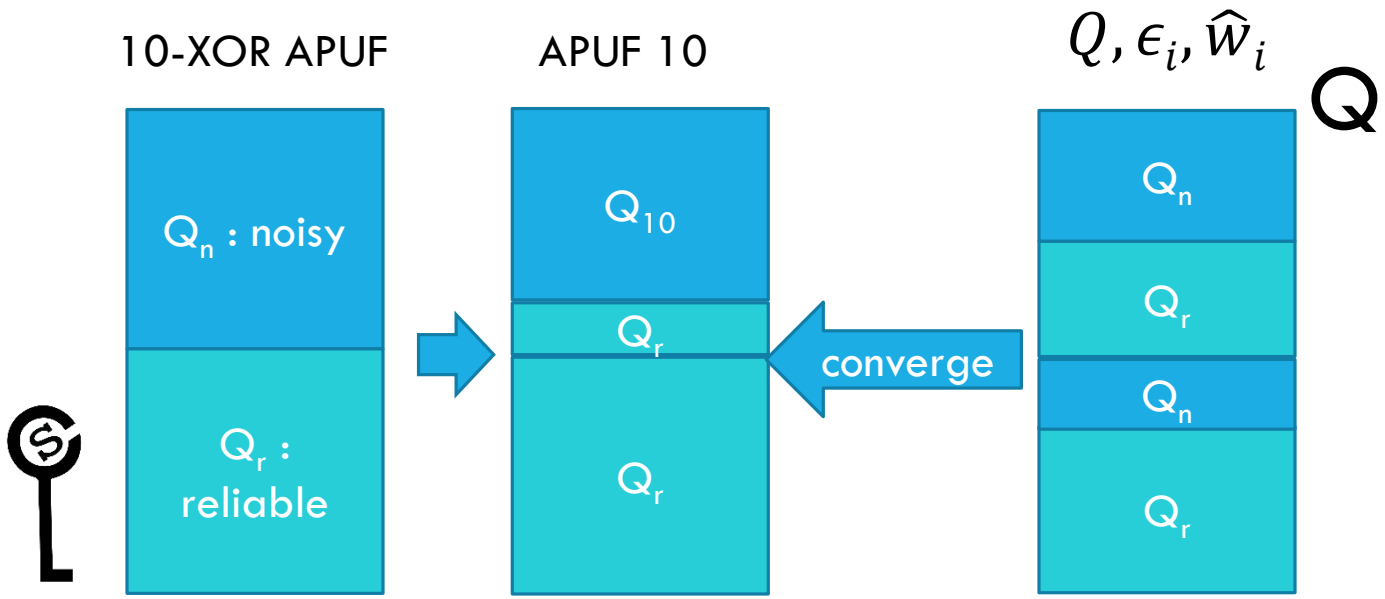
- (1) All the models $\widehat{w}_i$ in CMA-ES are models of APUF
- (2) $\widehat{w}_i$ can only converge to an APUF instance
- (3) CMA ES maximizes the matching Q of $\widehat{w}_i$ and Q of XOR APUF

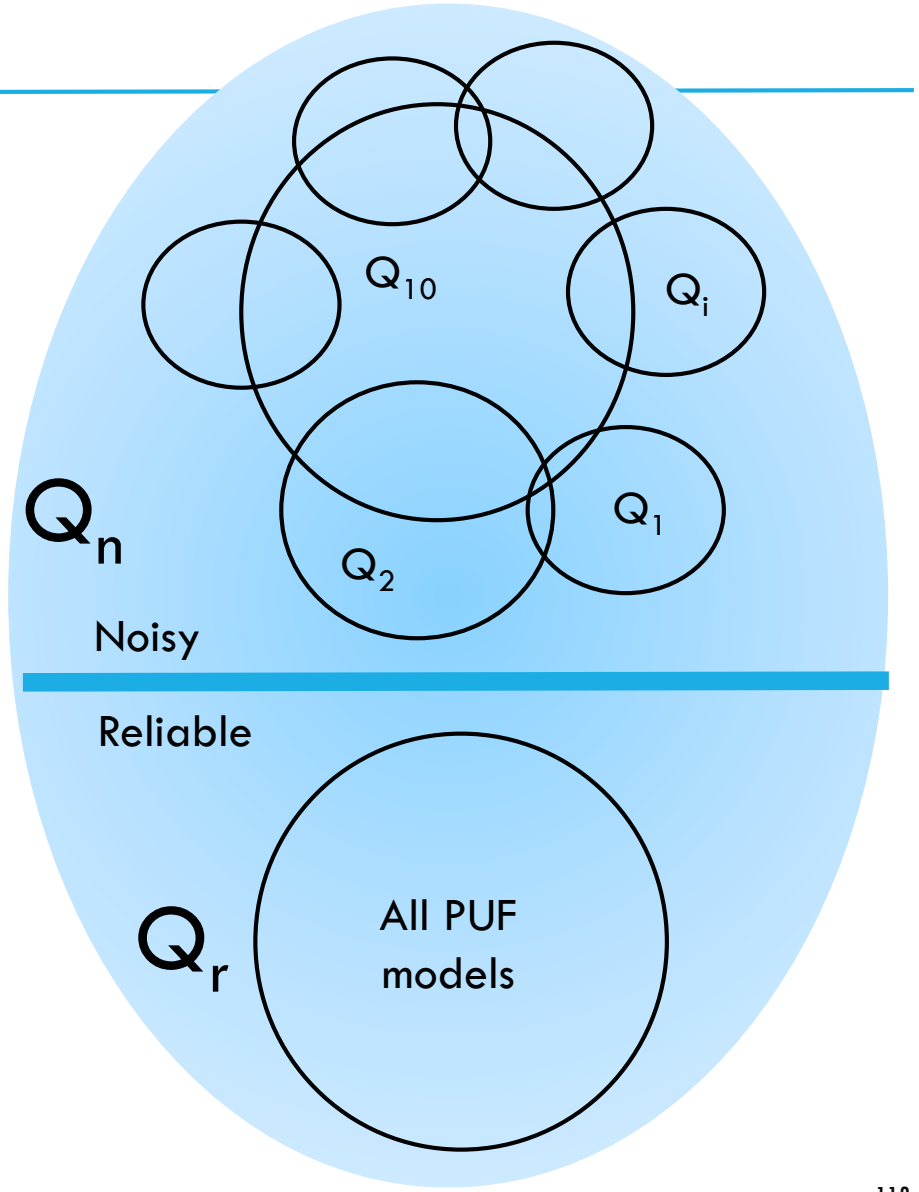




Key idea of the attack on XOR PUF

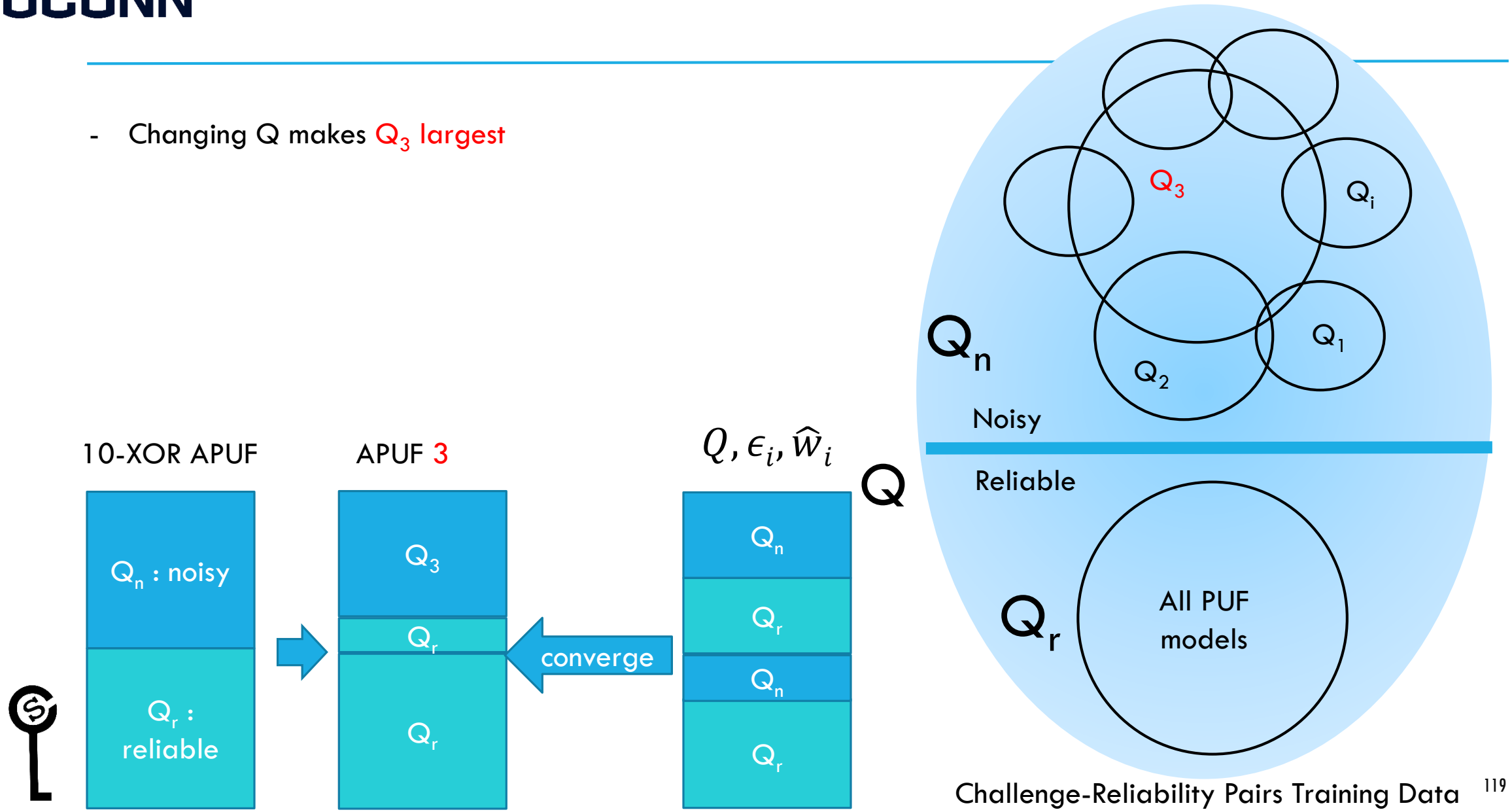
- (1) All the models $\widehat{w}_i$ in CMA-ES are models of APUF
- (2) $\widehat{w}_i$ can only converge to an APUF instance
- (3) CMA ES maximizes the matching Q of $\widehat{w}_i$ and Q of XOR APUF
- (1)+(2)+(3) CMA ES forces $\widehat{w}_i$ converges to APUF 10 because Q of APUF 10 is the representative of Q of XOR APUF.





Key idea of the attack on XOR PUF [5]

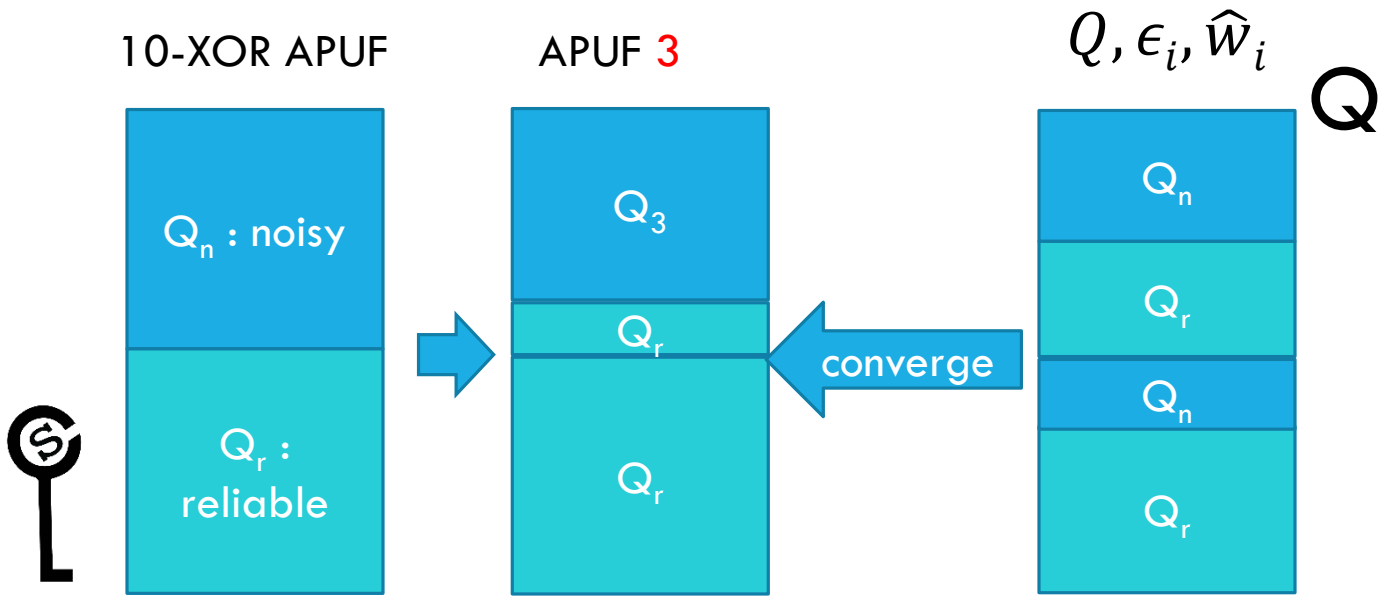
- Changing Q makes Q_3 largest

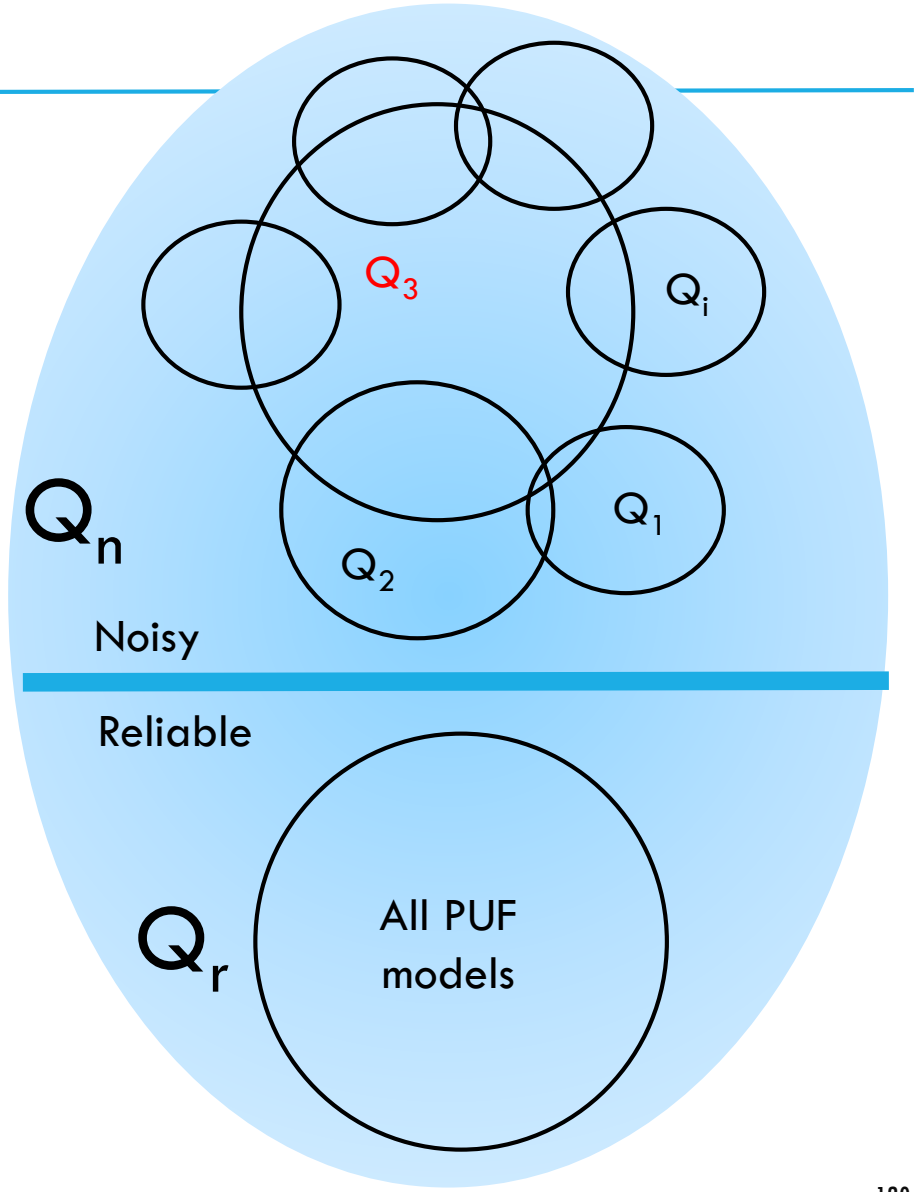


Key idea of the attack on XOR PUF

- Changing Q makes Q_3 largest

Keep change Q and apply CMA-ES attack on Q
to get models of all APUF instances



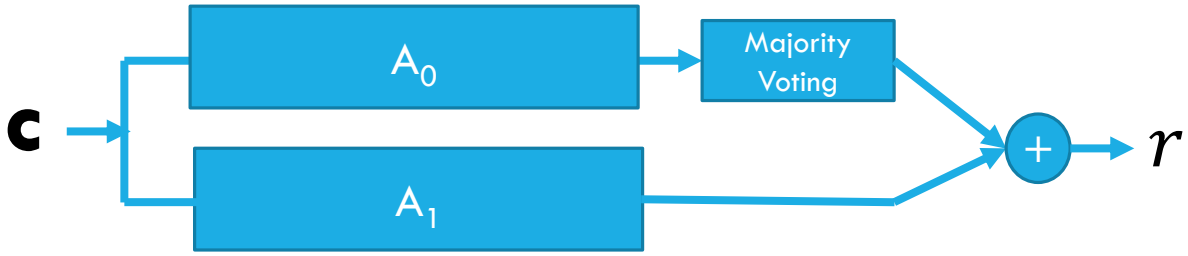


UCONN

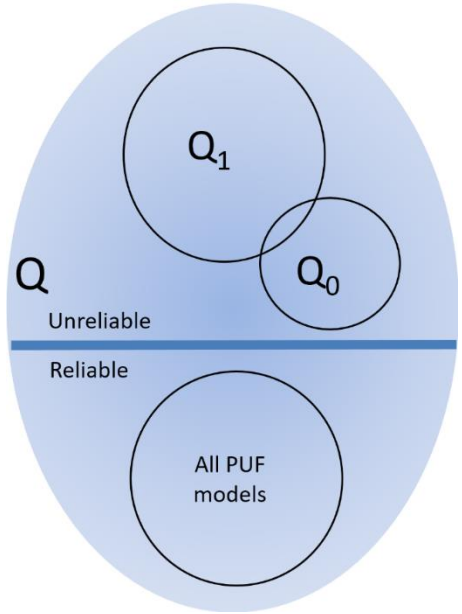
Question 2: How to make the attack on XOR PUF fail?

UCONN

Attack fails



2-XOR APUF with majority voting circuit at A_0



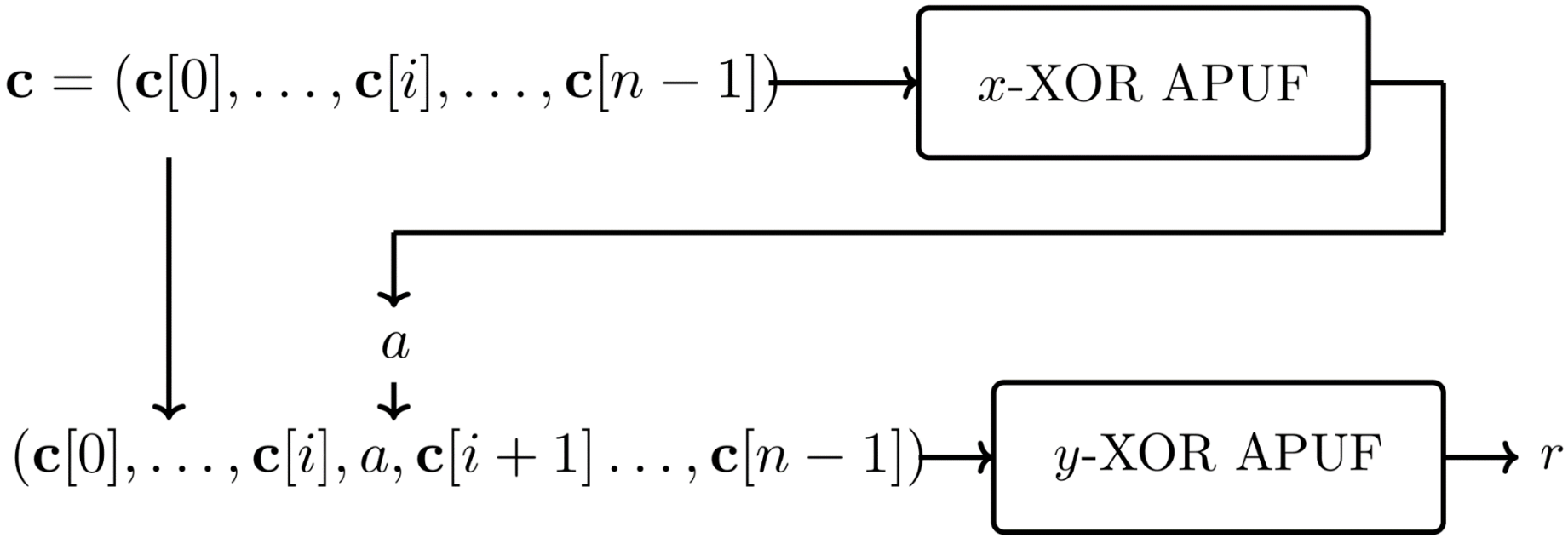
CMA ES never converges to APUF A_0 and always converges to APUF A_1 when majority voting mechanism in use.

UCONN

9. Becker Attack Resistance of iPUF

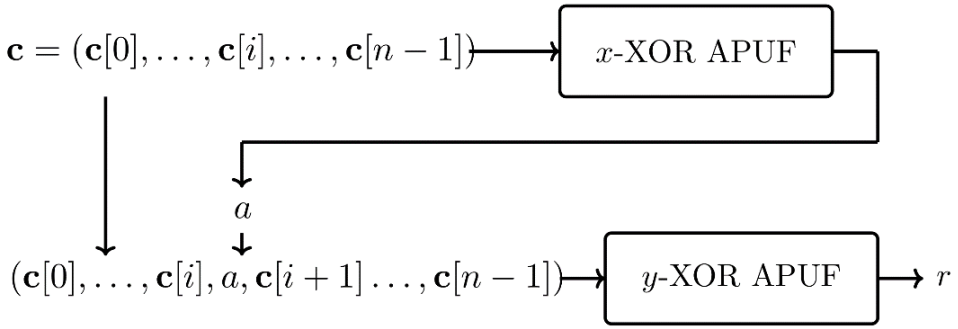
UCONN

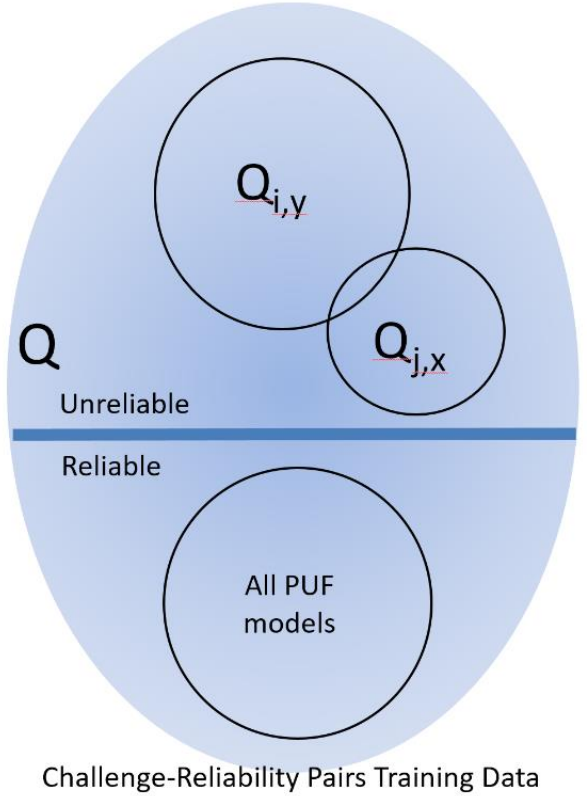
Interposed PUF (iPUF)



Security of iPUF wrt Reliability-based modeling attack

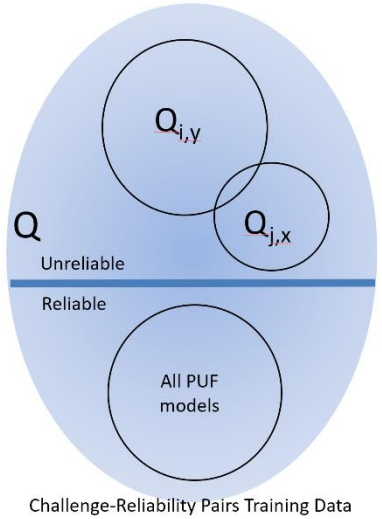
- Reason 1: the information of APUF instances in x-XOR PUF presented at the iPUF output is less compared to APUF instances in y-XOR PUF. Thus, the reliability based modeling attack never converges to any APUF instance in x-XOR PUF

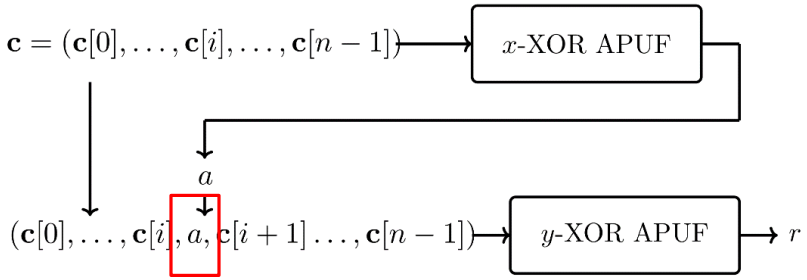




Security of iPUF wrt Reliability-based modeling attack

- Reason 2: to attack APUFs at x-XOR APUF, the adversary needs to compute Δ. But compute Δ is infeasible because the output of x-XOR PUF (a) is not known.





$Q, \epsilon_1, \widehat{w}_1$ Cannot compute $\Phi(c)$ *or* Δ

$Q \rightarrow$ *challenge* $c \rightarrow \Phi(c) \rightarrow \Delta = \widehat{w}_1 \cdot \Phi(c)$

$|\Delta| \leq \epsilon_1 \rightarrow$ *challenge c is noisy* → Q_n : noisy

$|\Delta| > \epsilon_1 \rightarrow$ *challenge c is reliable* → Q_r : reliable

$$\Delta = \Delta(n-1) = \mathbf{w} \cdot \mathbf{\Phi}^{\mathsf{T}}$$

$$\mathbf{\Phi}[i] = \prod_{j=i,\ldots,n-1} (1 - \mathbf{c}[j]), i = 0, \ldots, n-1$$

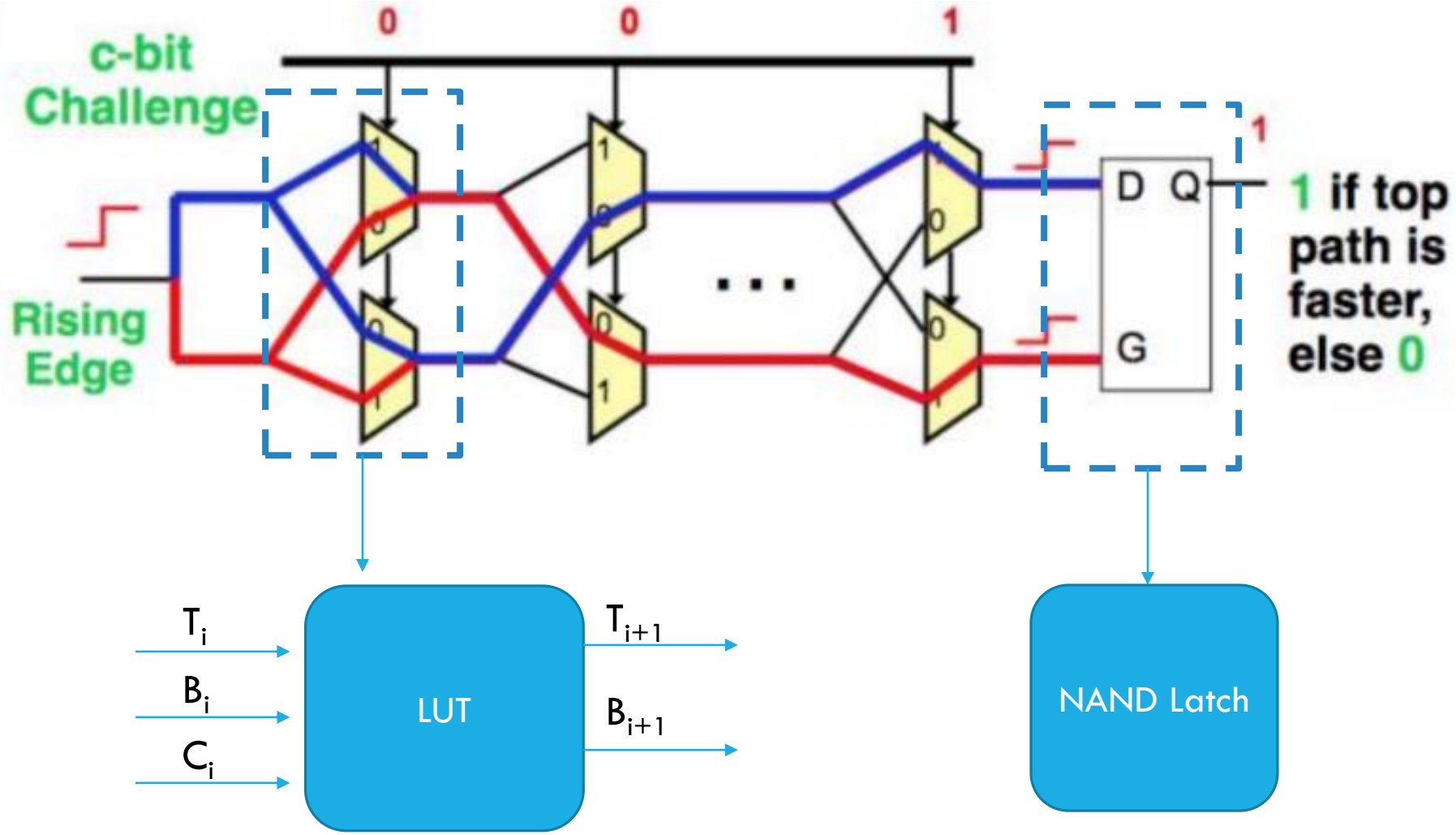
$$\mathbf{\Phi}[n] = 1$$

UCONN

10. Implementation

UCONN

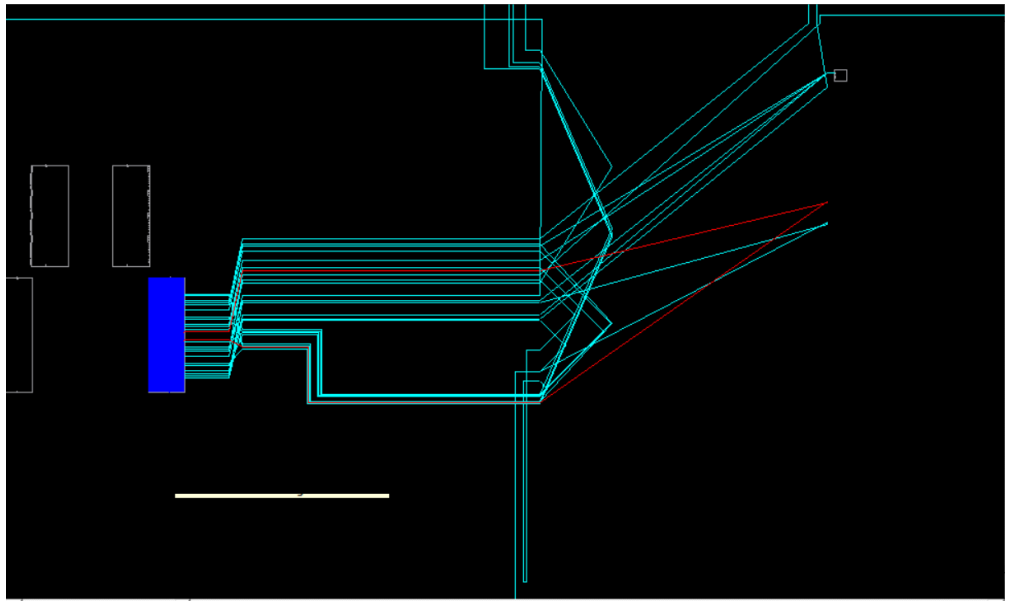
APUF Design



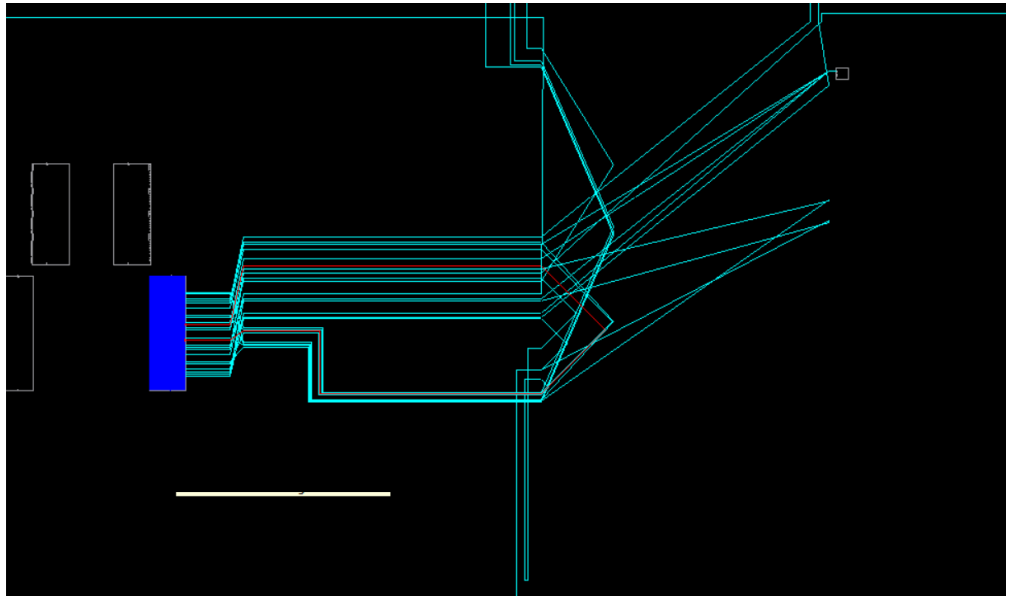
UCONN

Implementing APUFs on FPGAs

- The APUFs on FPGAs have very poor uniqueness (same APUF designs on an FPGA will create almost the same APUF instances)
  - Process variation in FPGAs does not play a big role in creating an APUF instance.
  - Routing delay variation in FPGAs, which is out of the control of an FPGA user, largely influences the created APUF instance.



Top Path



Bottom Path

UCONN

Solutions

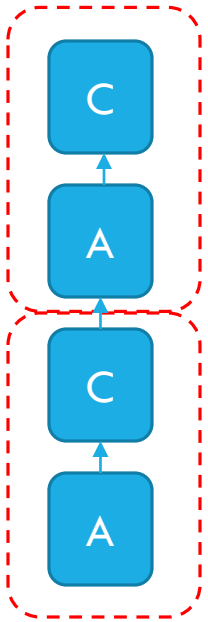
- Vary the APUF design slightly in every instance

1. Programmable Delay Line (PDL) – based APUF
   - Recent study shows a bias in the responses of PDL-based APUFs.
2. Vary the placement of switches in different APUF instances
   - Our choice
   - Place the switches in one column of an FPGA and vary the selection of the LUTs in the column

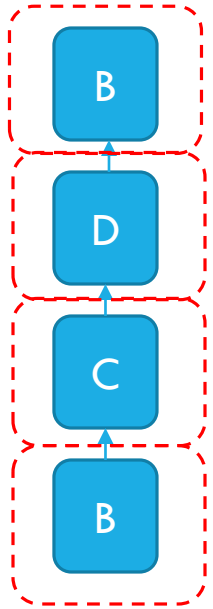
UCONN

Placement of Switches

- Pattern Placement
- Place the switches based on a pre-defined pattern (e.g., AC)



- Random Placement
- Place the switches in one randomly-selected LUT in each slice



UCONN

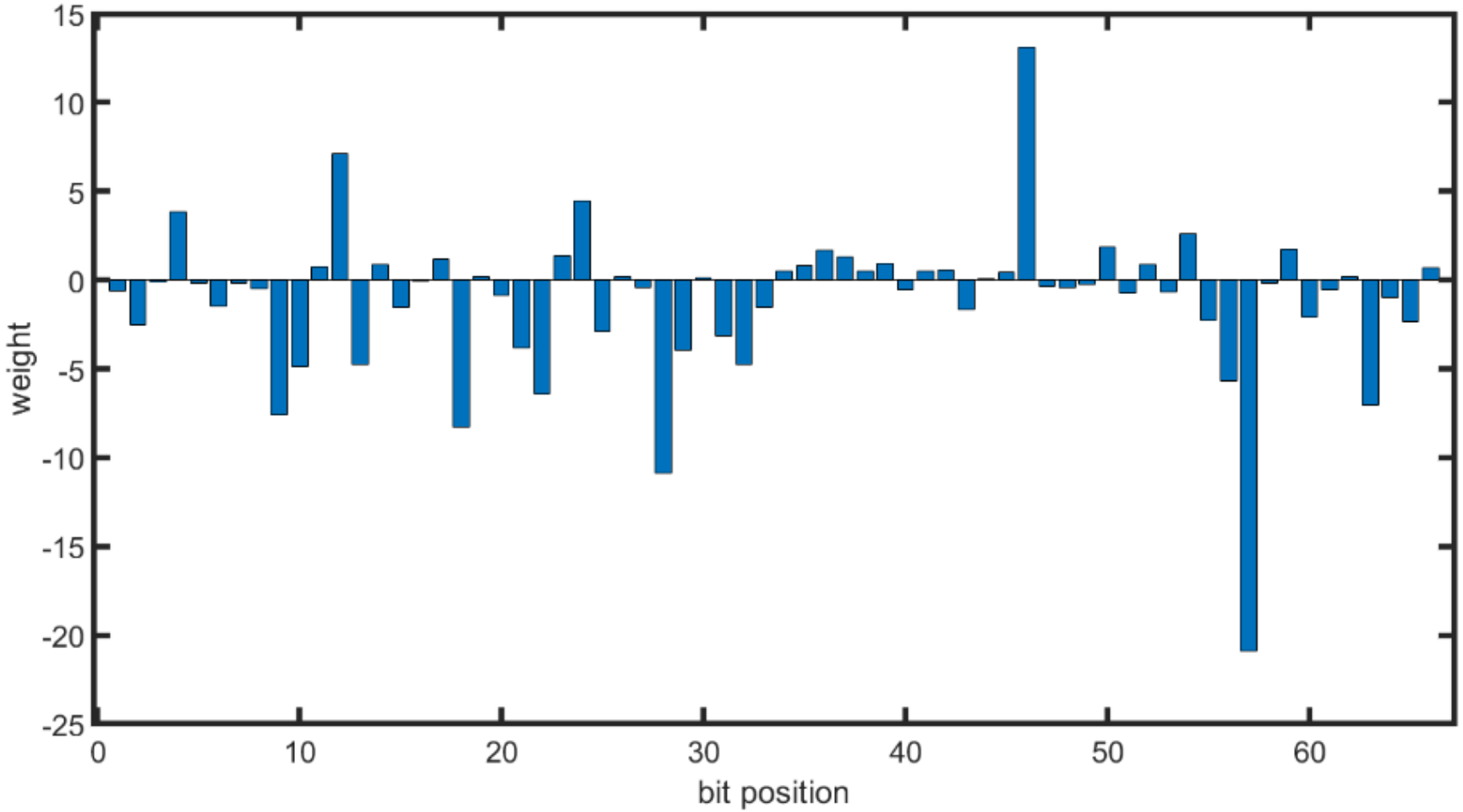
Placement of Switches

- Pattern Placement — Good Implementation
- Place the switches based on a pre-defined pattern (e.g., AC)
- Characterize the uniformity, uniqueness, and reliability before using the PUF instance.
- Pros:
  - The delay between each switch is more equally distributed.
  - The standard deviation of weights ranges from 1.11 to 3.77
- Cons:
  - Limited choices of patterns
  - One may not be able to build a sufficiently large XOR APUF.

- Random Placement — Bad Implementation
- Place the switches in one randomly-selected LUT in each slice
- Characterize the uniformity, uniqueness, and reliability before using the PUF instance.
- Pros:
  - Lots of choices of placement
  - A lot more unique PUF instances
- Cons:
  - Some delays may get significantly larger or smaller than the others.
  - It can sometimes lead to a successful attack on the PUF.
  - The standard deviation of weights ranges from 4.23 to 7.48

UCONN

Distribution of the weights of a badly implemented APUF



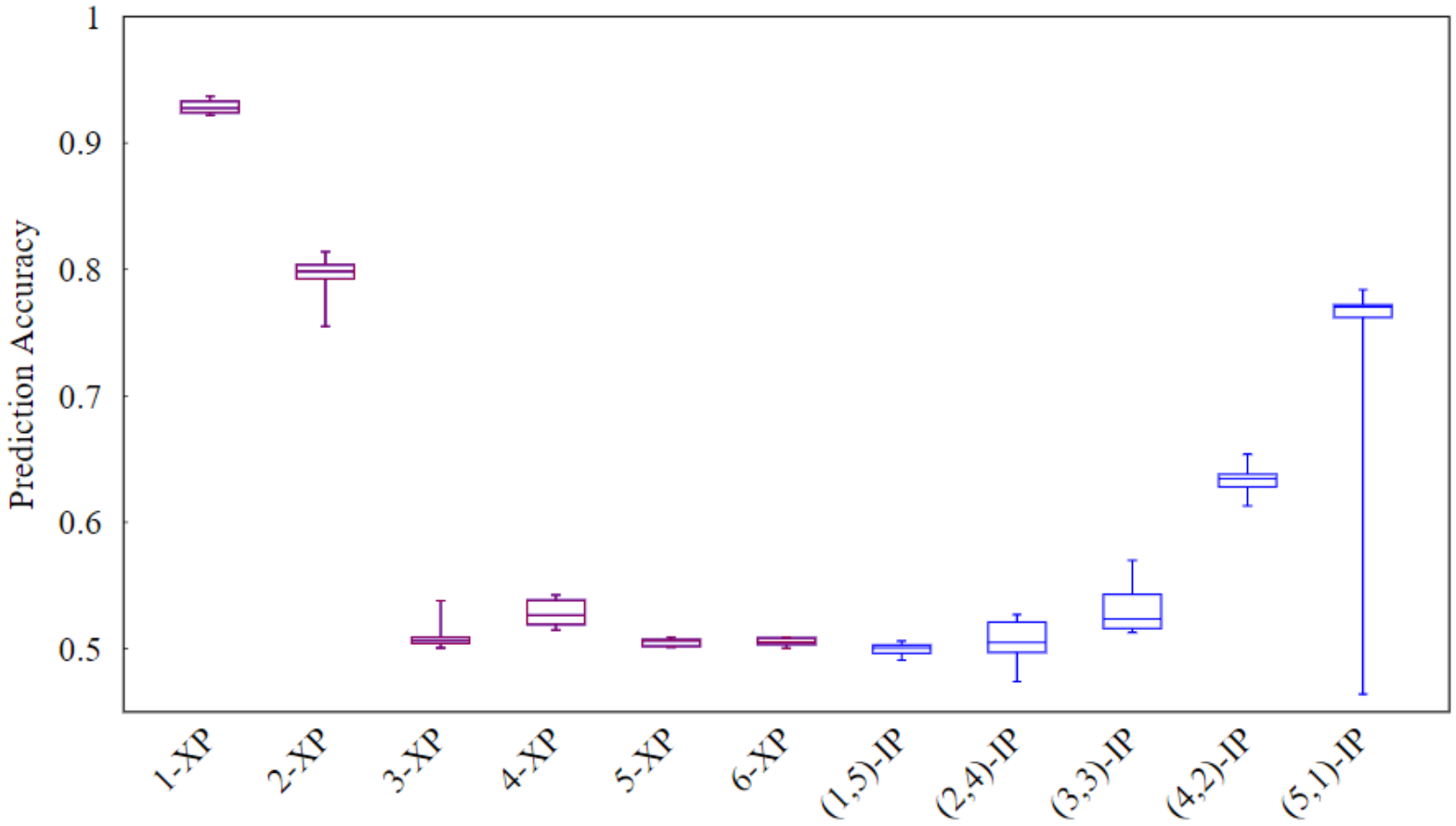
UCONN

Attacking Results on FPGA Implementations

Reliability-based attack on XOR APUFs on FPGA

	#CRPs used in one attack	Overall Noise Rate	Average Prediction Accuracy
2-XOR	50,000	1.44%	98.22%
3-XOR	90,000	2.38%	96.38%
4-XOR	140,000	2.92%	96.15%
5-XOR	200,000	3.80%	96.62%
6-XOR	260,000	4.47%	91.58%*

Model Complexity Comparison between iPUF and XOR APUF



$$(x, y) - IPUF \approx \left(y + \frac{x}{2}\right) - XOR\ PUF$$

UCONN

Reliability-based Attack on iPUF

- iPUF is secure against reliability-based attack by its design, so we used an (1,1)-iPUF in the experiments
- An (1,1)-iPUF with random placement (bad implementation) can be attacked by the reliability-based attack
  - The model of the upper layer APUF was discovered, then an attacker can carry on to attack the lower layer.
- An (1,1)-iPUF with pattern placement (good implementation) cannot be attacked by the reliability-based attack
  - No correct model was built

UCONN

Open Source

- All the PUF simulation, attacks, and FPGA implementation can be found at

  https://github.com/scluconn/DA_PUF_Library

UCONN

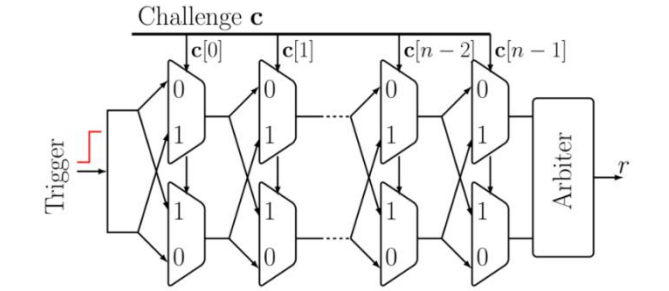
Other Contributions

- Enhanced Reliability based Modeling Attacks on APUF and XOR APUFs
- Prove why Logistic Regression on XOR APUF is the best attack and why Logistic Regression on iPUF is not applicable
- Implementation of iPUF and of modeling attacks on APUFs, XOR APUF, and iPUF
  https://github.com/scluconn/DA_PUF_Library/

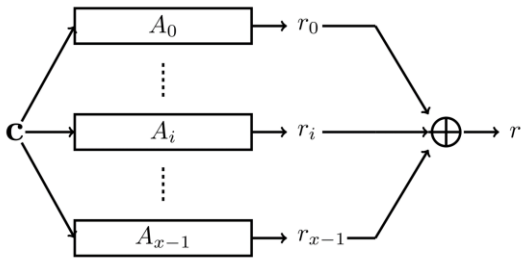
UCONN

Conclusion

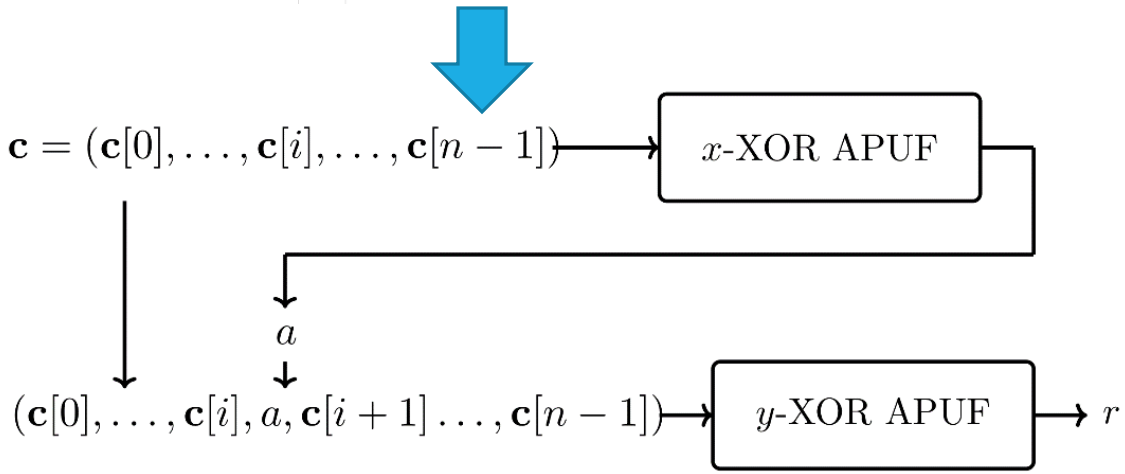
- We explain how the reliability-based modeling attack on XOR PUF works
- We propose a new lightweight PUF design (iPUF) which is secure against the state-of-the art of modelling attacks.



A n-stage classic Arbiter PUF with challenge $\mathbf{c} \in \{0,1\}^n$.



x-XOR PUF.



UCONN

Literature

1. https://slideplayer.com/slide/3927633/
2. Cryptanalysis of electrical PUFs via machine learning algorithms – Master Thesis of Jan Solter
3. The Gap Between Promise and Reality: On the Insecurity of XOR Arbiter PUFs CHES, September 16 th , 2015, Georg T. Becker
4. https://en.wikipedia.org/wiki/CMA-ES

Thank you for your attention!
and any questions?