PAC Learning of Arbiter PUFs

Fatemeh Ganji
Jean-Pierre Seifert
Shahin Tajik

TU Berlin, Berlin (Germany)

27. September 2014

SECT Security IN TELECOMMUNICATIONS

Agenda

Basics of arbiter PUFs, DFAs & PAC learning

- Why PAC learning of arbiter PUFs?

Representing arbiter PUFs by DFAs

- Discretization of arbiter PUFs
- Building a DFA out of an arbiter PUF

PAC learning of arbiter PUFs

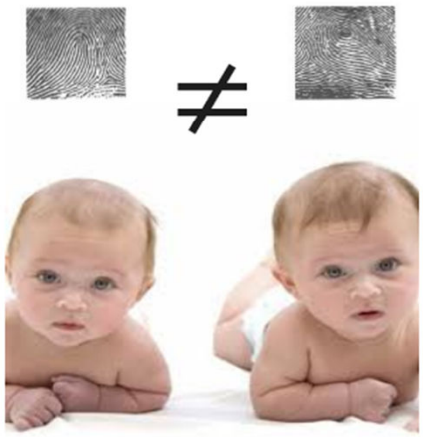
Conclusion

PUFs = Physically Unclonable Functions

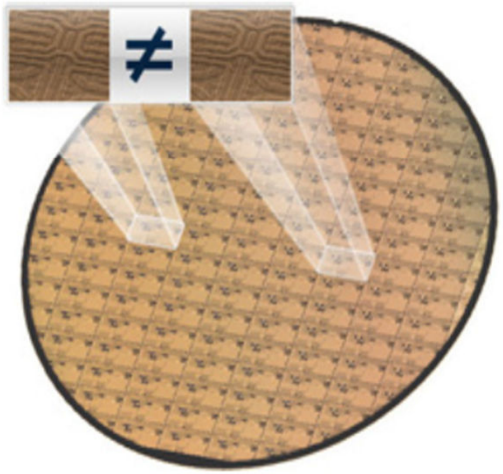
Silicon “Biometric” Technology

Physical Unclonable Function (PUF)

No two silicon chips, even with the same design, technology and fabrication process, are created alike. In the manufacturing process, there are unavoidable and uncontrollable variations at a molecular scale that make each silicon chip unique.



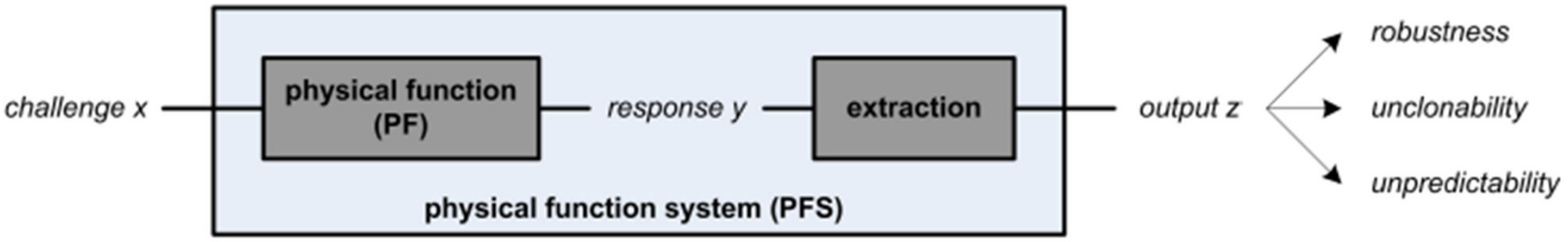
Each of us is unique and different …



… and so are silicon chips.

Formal Definition by „Strong PUFs and their pyhsical …“ @ ACM WESS 2013

A *Physical Function System* PFS is a probabilistic procedure which takes as input a challenge $x \in \mathcal{X}$ and generates an output $z \in \mathcal{Z}$



The Physical Function System is thus defined as $\mathsf{PFS}(x, h) \rightarrow (z, h')$ such that helper data is an empty string $h = \epsilon$ in setup mode, and the input helper data is returned unchanged $h = h'$ in reconstruction mode. The important properties of the above Physical Function System are formally defined as follows:

PFS = Physical Function System

Formal Definition by „Strong PUFs and their pyhsical …“ @ ACM WESS 2013

1. **Robustness**. Robustness of a PFS is represented by the probability that for a given PUF, the output generated by reconstruction phase matches the value generated in setup phase using its corresponding helper data h.
2. **Physical Unclonability**. Physical Unclonability relates to the probability that an adversary can build (or find) a PFS′ which shows the same behavior as another PFS. By the same behavior we mean that PFS′ generates the same output as PFS providing that PFS′ uses the helper data generated by PFS in setup mode.
3. **Unpredictability**. Unpredictability is another property of a Physical Function System which is important specially for strong PUFs. Roughly speaking, unpredictability is related to the probability that an adversary wins a predicting experiment such that he predicts the output corresponding to a new challenge from previously observed challenge-output pairs.

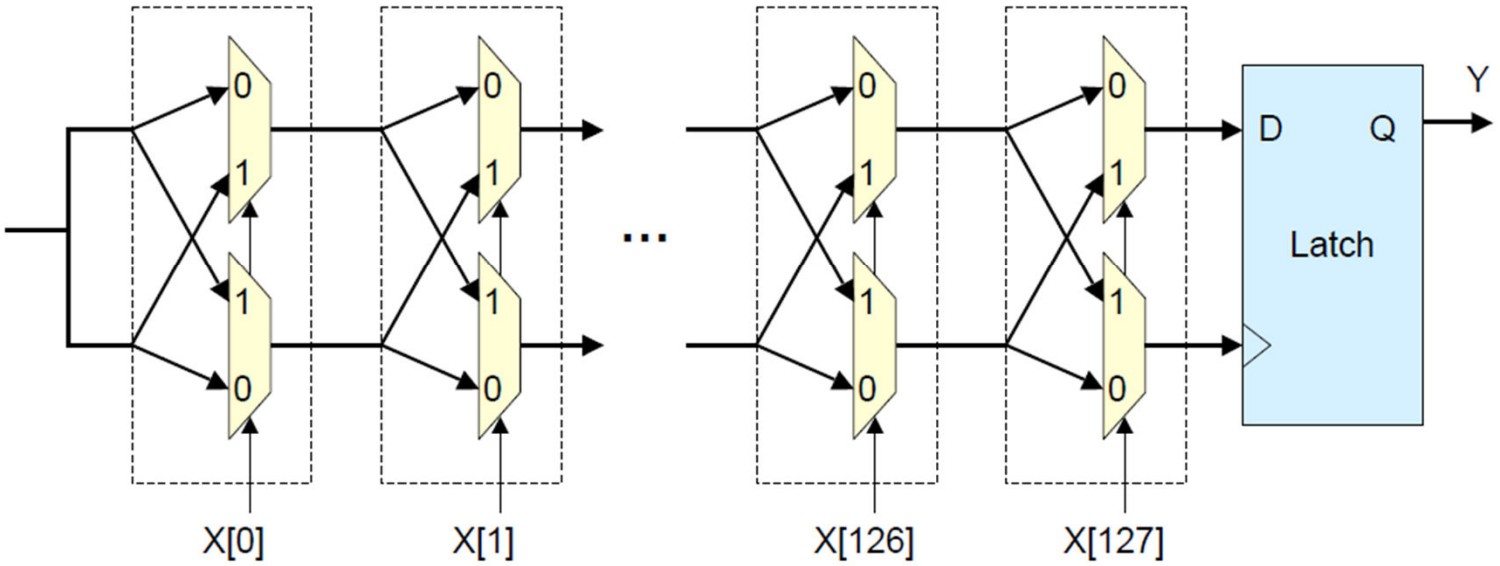
Our Definition

PUFs are physical input to output mappings, which are most often entangled with the intrinisc silicon properties of a chip. The input and output of a PUF are called *challenge* and *response*, respectively. A PUF can be described by the function $f_{\mathrm{PUF}} : \mathcal{C} \rightarrow \mathcal{Y}$ where $f_{\mathrm{PUF}}(c) = y$, $\mathcal{C} = \{0,1\}^n$ being the set of challenges, and $\mathcal{Y} = \{0,1\}$ the set of responses, c.f. [12]. PUFs in general have a set of crucial properties, c.f. [12].

12. Maes, R., Verbauwhede, I.: Physically Unclonable Functions: A Study on the State of the Art and Future Research Directions. In: Towards Hardware-Intrinsic Security, pp. 3–37. Springer (2010)

Verayo from US

Arbiter PUF

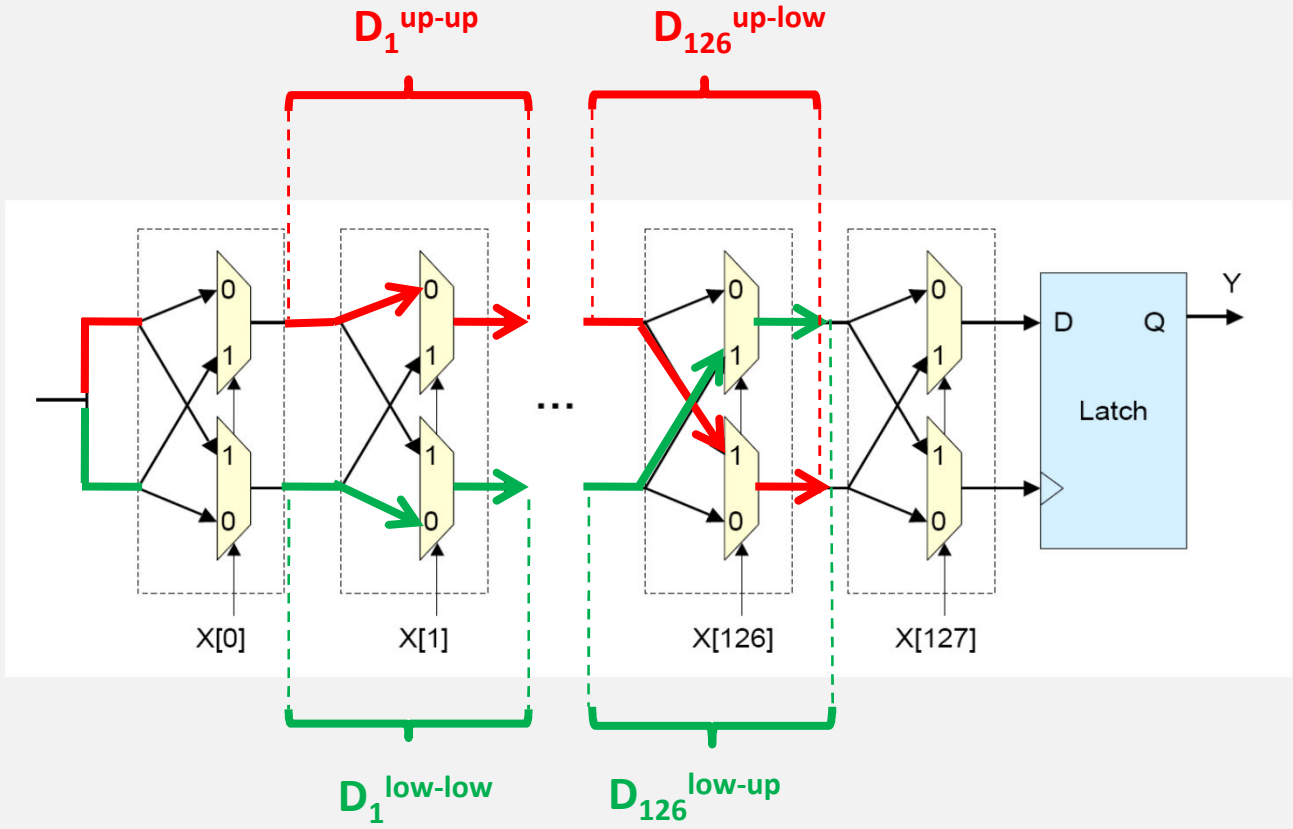


An arbiter PUF delay circuit. The circuit creates two delay paths with the same layout length for each input X, and produces an output Y based on which path is faster.

from G. Edward Suh, Srinivas Devadas: *Physical Unclonable Functions for Device Authentication and Secret Key Generation.* DAC 2007: 9-14.

Verayo from US

Arbiter PUF



from G. Edward Suh, Srinivas Devadas: *Physical Unclonable Functions for Device Authentication and Secret Key Generation.* DAC 2007: 9-14.

PAC learning

- Probably approximately correct learning (PAC learning) is a framework for mathematical analysis of machine learning.
- The learner receives samples and selects a generalization function (called the hypothesis) from a certain class of possible functions.
- The goal is that, with high probability (the "probably" part), the selected function will have low generalization error (the "approximately correct" part).
- The learner must be able to learn the concept given any arbitrary approximation ratio, probability of success, or distribution of the samples.

PAC learning

- Alg. is given samples $S = \{\ldots, (x, y), \ldots\}$ presumed to be drawn from some distribution D over instance space $X = \{0, 1\}^n$, labeled by some target function f.
- Alg. does optimization over S to produce some hypothesis h.
- Goal is for h to be close to f over D.
- Allow failure with small prob. δ (to allow for chance that S is not representative).

PAC learning

- **A concept class is a set of functions, together with a representation of them.**
- **Alg. A PAC-learns concept class C by hypothesis class H if for any target *f* in C, any dist. D over X, any $\varepsilon, \delta > 0$,**
  - A uses at most poly(n, $1/\varepsilon$, $1/\delta$, size(f)) examples and running time.
  - With probability $1-\delta$, A produces h in H of error at most ε.

accuracy

confidence

Agenda

Basics of arbiter PUFs, DFAs & PAC learning

- **Why PAC learning of arbiter PUFs?**

Representing arbiter PUFs by DFAs

- Discretization of arbiter PUFs
- Building a DFA out of an arbiter PUF

PAC learning of arbiter PUFs

Conclusions

PUF Modeling Attacks: An Introduction and Overview

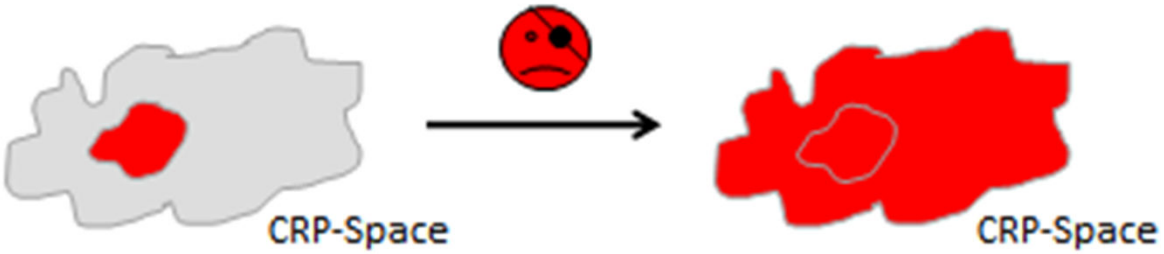
DATE 2014
Dresden, March 27, 2014

Ulrich Rührmair [1], Jan Sölter [2]

[1] TU München, Germany
[2] FU Berlin, Germany

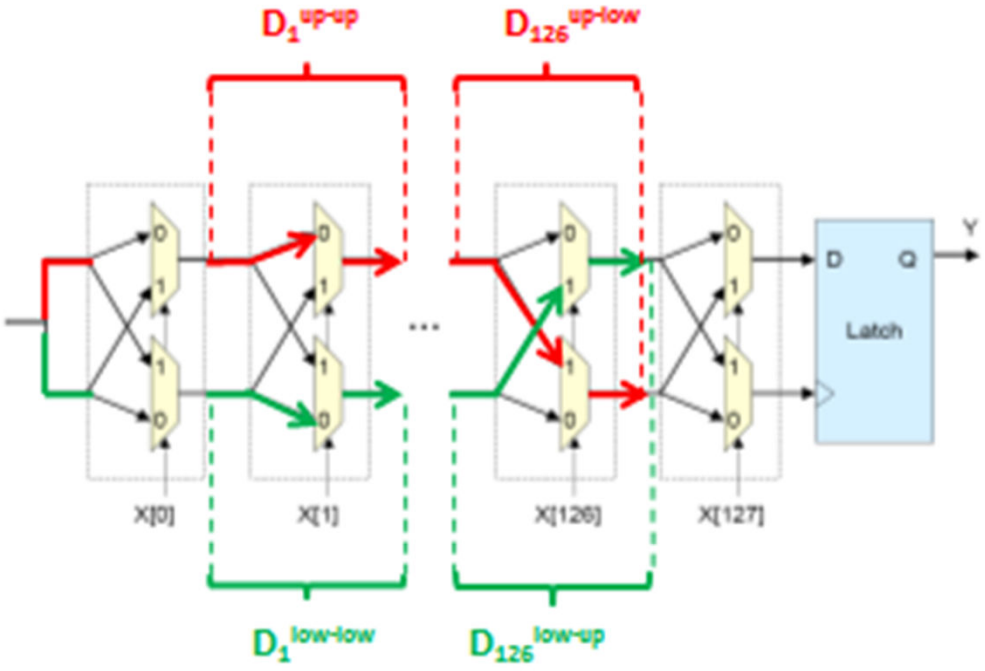
Steps in a Modeling Attack

- Adversary collects a **subset** of all CRPs
- **Derives a simulation algorithm** that predicts or extrapolates the PUF-behavior on all CRPs



- Weapon of choice: **Machine learning (ML) techniques!**
  - Typical supervised ML problem…
- Usually, certain assumptions about the **internal functionality** of the attacked PUF are made to speed up ML
  - Some ***„internal model"*** of the PUF is assumed

Arbiter PUFs and Their Internal Model: Linear Additive Delay Model (LADM)



- LADM: Overall delay = sum of internal delays in subcomponents

Linear Additive Delay Model Cont.

- LADM in mathematical terms:

$$R(C_i) = \text{Func}\,(\underbrace{D_1^{\text{up-up}}, \ldots, D_{128}^{\text{low-low}}}_{\mathbf{D}}, C_i)$$

D (with dim(**D**) = 4x128 = 512)

- MA tries to derive **D** from known subset of CRPs

Results for Arbiter PUFs on FPGA/ASIC Data

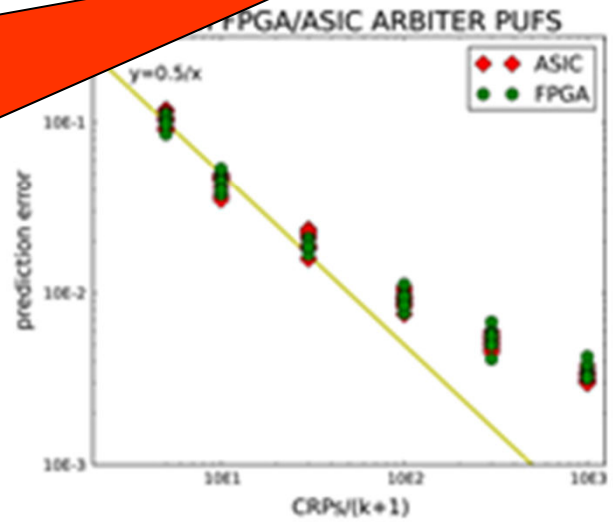
TABLE IX
LR ON ARB PUFS OF BITLENGTH 64 FOR FPGA AND ASIC DATA, COLLECTED UNDER STABLE TEMPERATURE AND MAJORITY VOTING

ML Method	CRP Source	Prediction Rate	CRPs	Training Time
LR	FPGA	> 95% > 99%	650 6500	0.12 sec 0.83 sec
LR	ASIC	> 95% > 99%	650 6500	0.11 sec 0.76 sec

Rührmair et al.,
IEEE T-IFS, 2013

Heuristic Results!

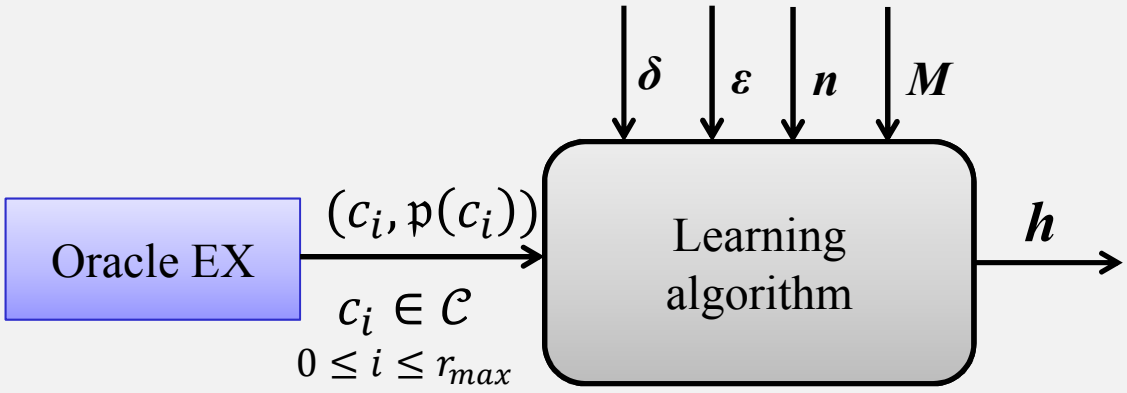
Not at all a proper mathematical proof of Learnability.



Main Theorem

For the class of arbiter PUFs of length *n* and some (PUF inherent) value *M* we will prove the existence of a PAC learner $\mathcal{L}$ with:

Theorem 1. Let $N := O\ (nM^2)$ that represents the number f live states, then **L** returns a hypothesis *h* after at most $O(N + (\frac{1}{\varepsilon})(N \log\left(\frac{1}{\delta}\right) + N^2))$ calls to *EX*, and with probability at least $(1 - \frac{\delta}{2})$, h is an ε/2-approximation of S^1



A labeled example is the pair $(c, \mathfrak{p}(c))$ and the set of positive examples (i.e., $\mathfrak{p}(c) = 1$) is denoted by S^1, whereas the set of negative examples is S^0.

Agenda

Basics of arbiter PUFs, DFAs & PAC learning

- Why PAC learning of arbiter PUFs?

Representing arbiter PUFs by DFAs

- Discretization of arbiter PUFs
- Building a DFA out of an arbiter PUF

PAC learning of arbiter PUFs

Conclusions

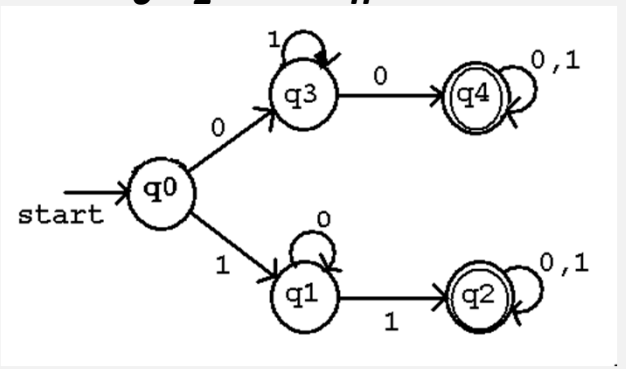
DFAs

A deterministic finite automaton M is a 5-tuple, $(Q, \Sigma, \delta, q_0, F)$, consisting of

- a finite set of states (Q)
- a finite set of input symbols called the alphabet (Σ)
- a transition function ($\delta : Q \times \Sigma \rightarrow Q$)
- a start state ($q_0 \in Q$)
- a set of accept states ($F \subseteq Q$)

Let $w = a_1 a_2 \ldots a_n$ be a string over the alphabet Σ. The automaton M accepts the string w if a sequence of states, $r_0, r_1, \ldots, r_n$, exists in Q with the following conditions:

1. $r_0 = q_0$
2. $r_{i+1} = \delta(r_i, a_{i+1})$, for $i = 0, \ldots, n-1$
3. $r_n \in F$.



Agenda

Basics of arbiter PUFs, DFAs & PAC learning

- Why PAC learning of arbiter PUFs?

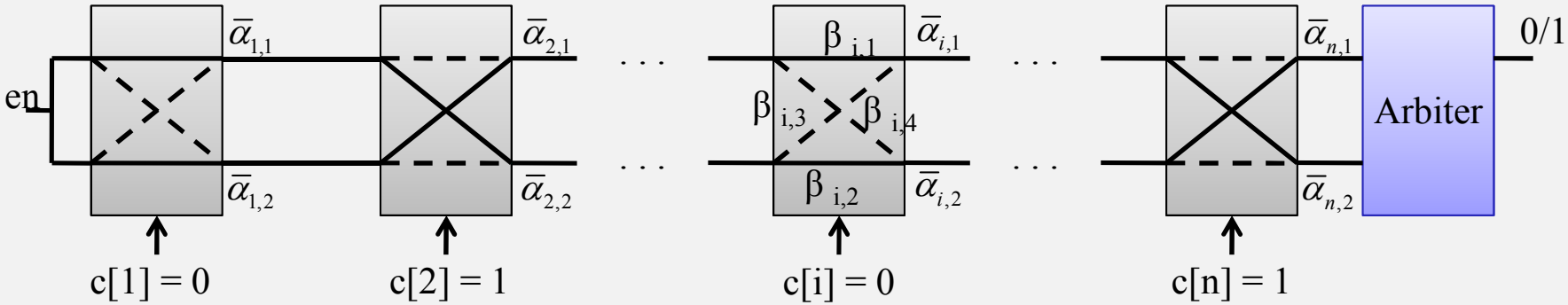
Representing arbiter PUFs by DFAs

- **Discretization of the delays of an arbiter PUF**
- Building a DFA out of an arbiter PUF

PAC learning of arbiter PUFs

Conclusions

Discretization of the delay of an arbiter PUF



$$B_i \sim N(\mu_i, \omega_i)$$

$$A_i = \sum_{k=1}^{i} B_k$$

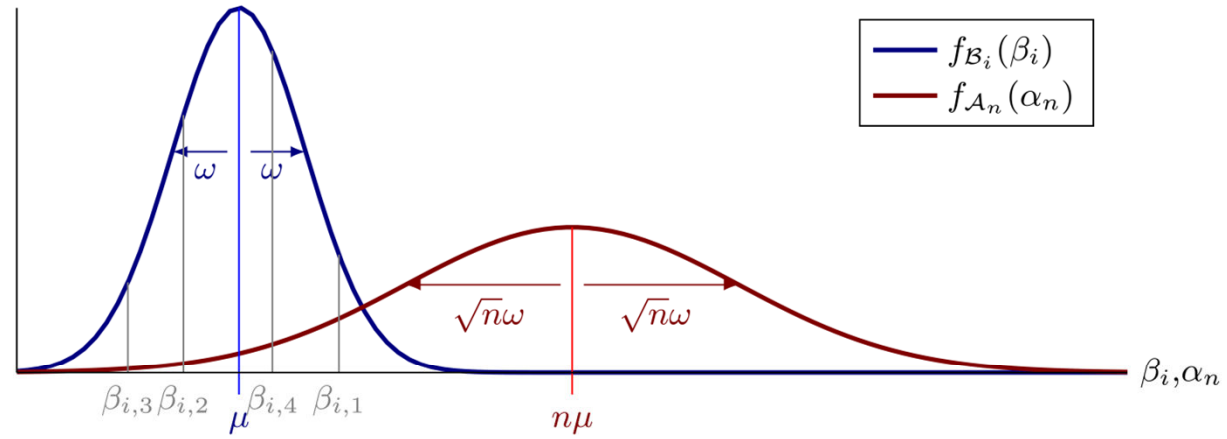


Fig. 2: The distribution of $\mathcal{B}_i$ (blue) with the mean μ and deviation ω. Four examples of possible realization of $\mathcal{B}_i$ are $\beta_{i,1}$, $\beta_{i,2}$, $\beta_{i,3}$, and $\beta_{i,4}$, which correspond to four delays at the i^{th} stage. The distribution of total propagation delay from the enable point to the outputs of last stage in an arbiter PUF with length n (red), with mean $n\mu$ and deviation $\sqrt{n}\omega$.

Discretization of the delays of an arbiter PUF

- $\mathcal{A}_n$ corresponds to the total propagation delay at the last stage and will have the mean $n\,\mu$ and the standard deviation $\omega\sqrt{n}$.

- Thus, all statistically relevant delay values lie whp. in a limited interval, whose length is $6\omega\sqrt{n}$ (i.e. within three standard deviation away from the mean value $n\,\mu$).

- Now, the arbiter at the end of the chain has a limited precision γ for comparing the total propagation delays of two paths. Hence, it can only compare two signals with a delay above the threshold γ.

Discretization of the delays of an arbiter PUF

- **Definition of a mapping f: $\mathbb{R} \rightarrow \mathbb{Z}$:**

$$\forall \bar{\alpha} \in [n\mu - 3\omega\sqrt{n}, n\mu + 3\omega\sqrt{n}]: f(\bar{\alpha}) = \left\lfloor \frac{\bar{\alpha} - (n\mu - 3\omega\sqrt{n})}{\gamma} \right\rfloor$$

- **As a result, all real delay values lying within three standard deviation away from the mean value $n\mu$ are mapped to integer values, ranging from 0 to**

$$M = \left\lfloor \frac{6\omega\sqrt{n}}{\gamma} \right\rfloor$$

- **The number of delay values, which can be observed and compared by the arbiter is limited to *M+1***

Discretization of the delays of an arbiter PUF

- **Mapping of real-valued $\bar{\alpha}_{i,j}$ to integer values ($\alpha_{i,j}$) lying between 0 and M**
  - corresponding to the minimum and maximum real values

$$\alpha_{i,j} \in Z, \qquad \alpha_{i,j} \in [0, M]$$

- **Response of the arbiter**
  - "1" if $\alpha_{n,1} - \alpha_{n,2} \geq 1$
  - "0" if $\alpha_{n,2} - \alpha_{n,1} \geq 1$
  - metastable condition, if $|\alpha_{n,1} - \alpha_{n,2}| = 0$.

Agenda

Basics of arbiter PUFs, DFAs & PAC learning

- Why PAC learning of arbiter PUFs?

Representing arbiter PUFs by DFAs

- Discretization of arbiter PUFs
- **Building a DFA out of an arbiter PUF**

PAC learning of arbiter PUFs

Conclusions

Building a DFA out of an arbiter PUF

- challenge-response functionality of a PUF [1]
  - the mapping $f_{PUF}: \mathcal{C} \rightarrow \mathcal{Y}$
  - $f_{PUF}(c) = y$, $\mathcal{C}$ is the set of challenges, and $\mathcal{Y}$ is the set of responses, $\mathcal{C} = \{0,1\}^n$ and $\mathcal{Y} = \{0,1\}$.

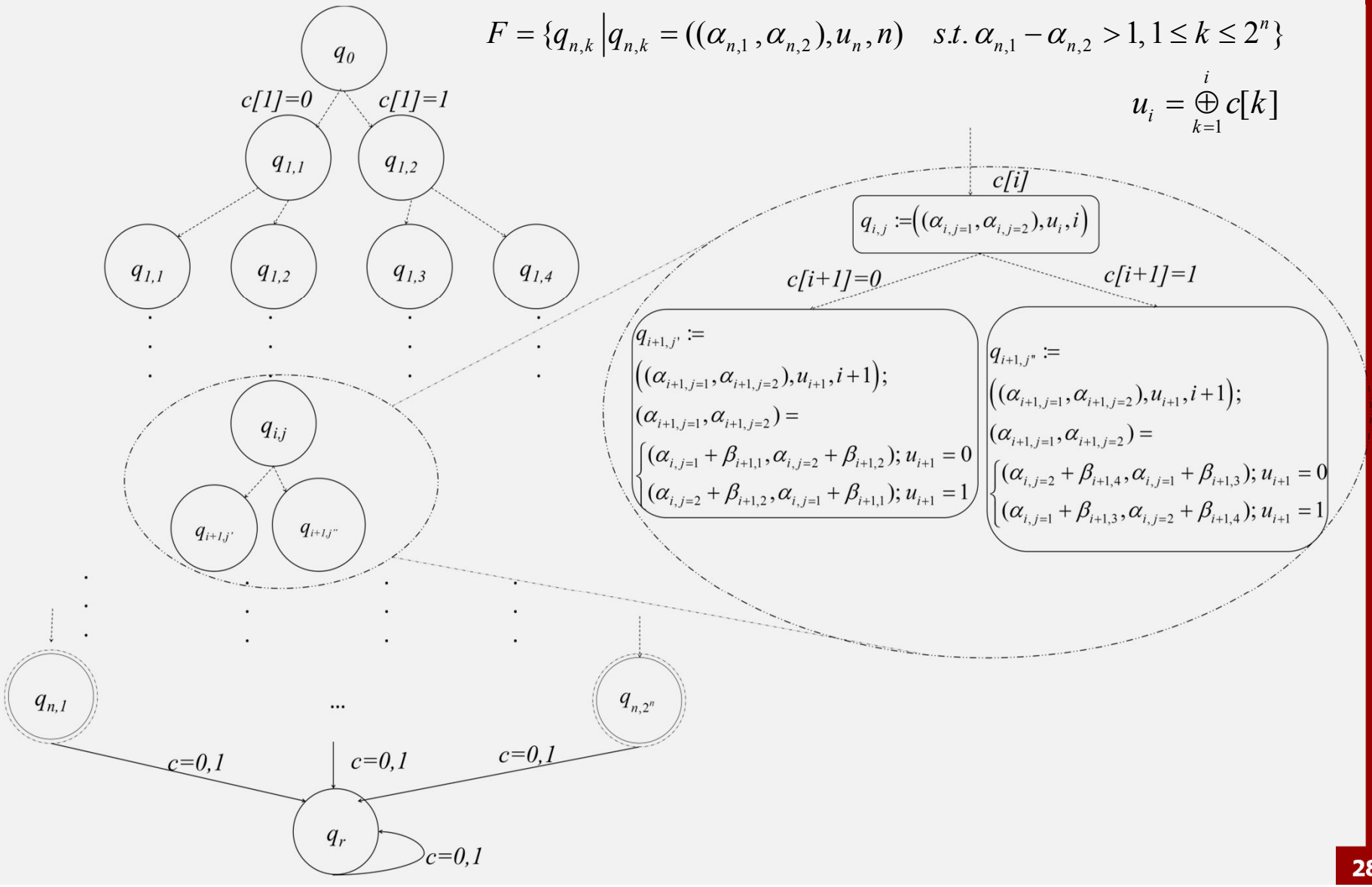
- For an arbiter PUF:
  - $\mathcal{C} = \{0,1\}^n$ and $\mathcal{Y} = \{0,1\}$. Let us $L_{PUF} := \{c \in C | f_{PUF}(c) = 1\}$
  - $L_{PUF} \subseteq \{0,1\}^n \subseteq \Sigma^*$, where Σ={0,1}
  - L_{PUF} : the accepted language of a certain automaton that accepts those strings $c \in C$, whose length is n and $f_{PUF}(c) = 1$

[1] Maes, R., Verbauwhede, I.: Physically Unclonable Functions: A Study on the State of the Art and Future Research Directions. In: Towards Hardware-Intrinsic Security, pp. 3{37. Springer (2010)

Building a DFA out of an arbiter PUF

$$F = \{q_{n,k} \,|\, q_{n,k} = ((\alpha_{n,1}, \alpha_{n,2}), u_n, n) \quad s.t.\ \alpha_{n,1} - \alpha_{n,2} > 1, 1 \le k \le 2^n\}$$

$$u_i = \bigoplus_{k=1}^{i} c[k]$$



Collisions

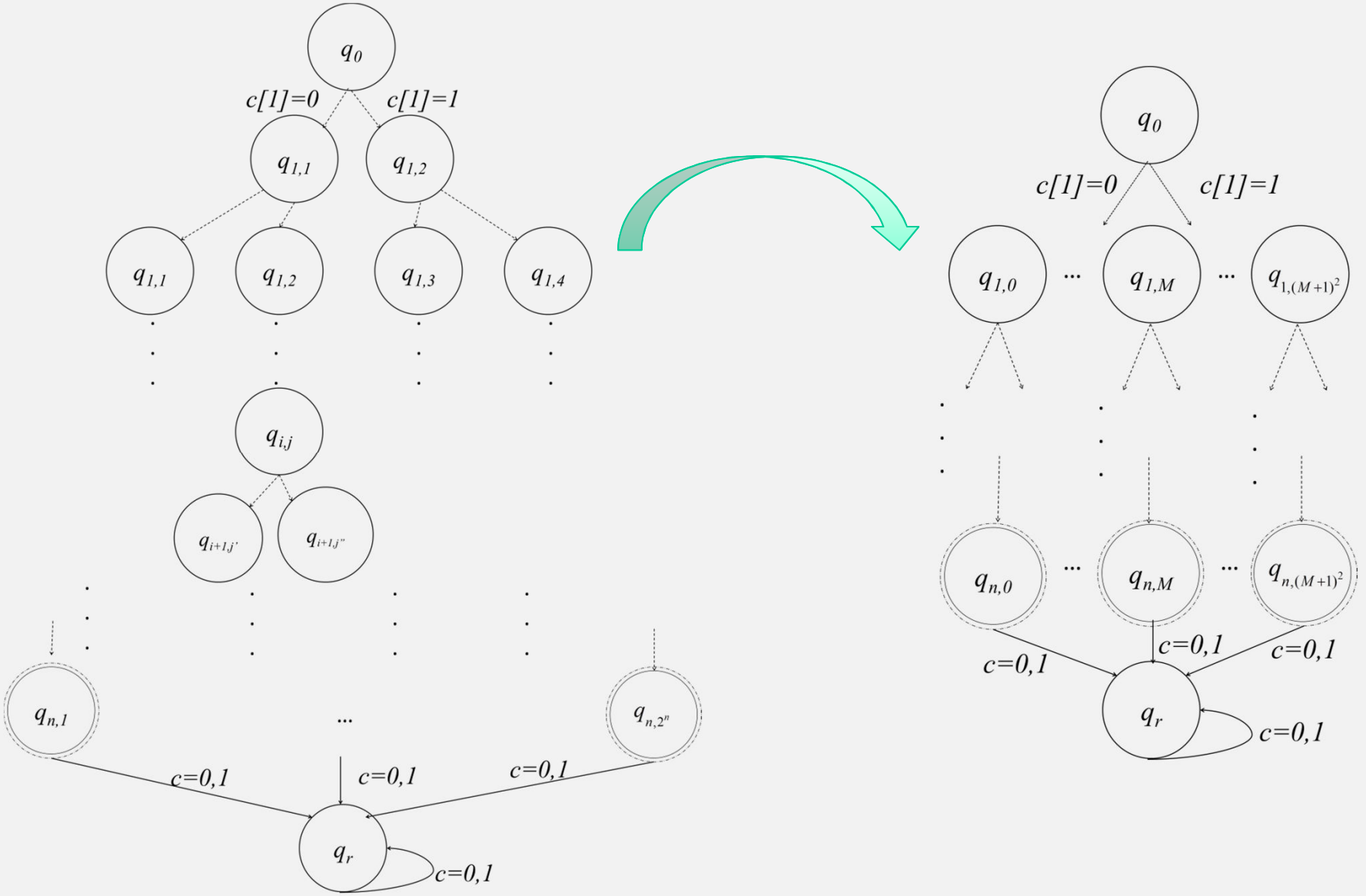
- **Size of the DFA based upon $\alpha_{i,j}$: exponential in n**
  - This representation of an arbiter PUF cannot be learned in polynomial time at all! BUT…

The total delay values can be mapped to [0,M]

- **The number of possible values of $\alpha_{i,j}$ $(1 \leq i \leq n\,, 1 \leq j \leq 2)$: $M + 1$**
  - The number of nodes in each level: $(M + 1)^2$
  - Total number of distinguishable states is limited by

$$O(n(M + 1)^2)$$

Shrunk DFA representing an arbiter PUF



Agenda

Basics of arbiter PUFs, DFAs & PAC learning

- Why PAC learning of arbiter PUFs?

Representing arbiter PUFs by DFAs

- Discretization of arbiter PUFs
- Building a DFA out of an arbiter PUF

PAC learning of arbiter PUFs

Conclusions

Learning an unknown regular language from *EX*

- **For the polynomial-size DFA we can now describe a PAC-learning algorithm to efficiently learn the challenge-response behavior of a given arbiter PUF.**
- **A PAC learner for DFAs is given by Dana Angluin**

THEOREM 7. *If n is the number of states in the minimum dfa for the unknown regular set U, then L_a^* terminates after $O(n+(1/\varepsilon)(n\log(1/\delta)+n^2))$ calls to the $EX(\ )$ oracle. Moreover, the probability that the acceptor output by L_a^* is an ε-approximation of U is at least $1-\delta$.*

INFORMATION AND COMPUTATION **75**, 87–106 (1987)

- **A given PUF provides the learner with access to the Oracle $EX := f_{PUF}$.**

Value *M*

- The maximum delay deviation of each inverter used in the PUF chain: 9 ps [1]
  - For both cases, i.e., direct and crossed paths on average for a Xilinx Virtex-5 FPGA
  - For XC5VLX110 chips (Xilinx Virtex-5 family) Delay deviation: smaller than 10 ps [2]
- Let $6\omega = 10\ ps, \mathrm{n} = 128, \gamma = 2.5\ ps$ then

$$M = \left\lceil \frac{6\omega\sqrt{n}}{\gamma} \right\rceil = 46 \quad \Rightarrow \quad size\ (A) = 282,752 \ll O(2^{128})$$

[1] Mahmoud, A., Rührmair, U., Majzoobi, M., Koushanfar, F.: Combined Modeling and Side Channel Attacks on Strong PUFs. Tech. rep., Cryptology ePrint Archive: Report 2013/632, 2013, https://eprint. iacr. org/2013/632 (2013).
[2] Majzoobi, M., Koushanfar, F., Devadas, S.: FPGA PUF Using Programmable Delay lines. In: Information Forensics and Security (WIFS), 2010 IEEE International Workshop on. pp. 1{6. IEEE (2010).

Agenda

Basics of arbiter PUFs, DFAs & PAC learning

- Why PAC learning of arbiter PUFs?

Representing arbiter PUFs by DFAs

- Discretization of arbiter PUFs
- Building a DFA out of an arbiter PUF

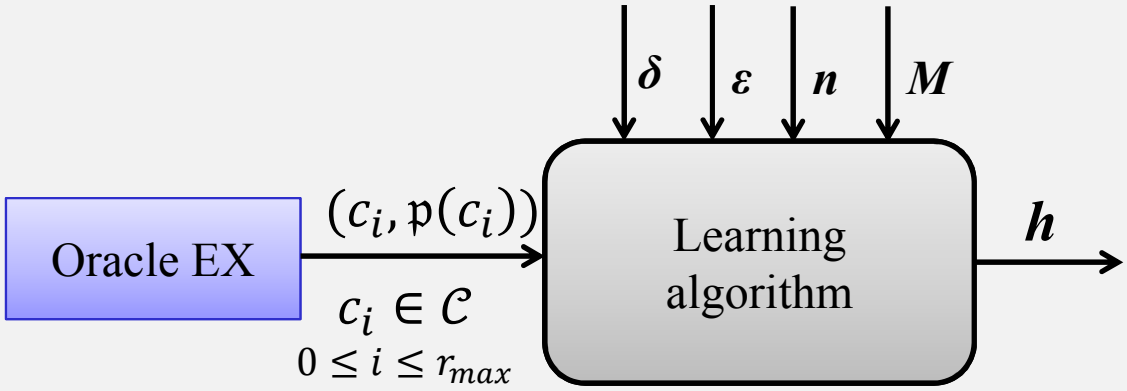
PAC learning of arbiter PUFs

Conclusion

Conclusion

For the class of arbiter PUFs of length *n* and some (PUF inherent) value *M* the following holds:

Theorem 1. Let $N := O\ (nM^2)$ that represents the number f live states, then **L** returns a hypothesis *h* after at most $O(N + (\frac{1}{\varepsilon})(N \log\left(\frac{1}{\delta}\right) + N^2))$ calls to *EX*, and with probability at least $(1 - \frac{\delta}{2})$, h is an ε/2-approximation of S^1



Thank you for your attention!

Questions?