6.857 L16

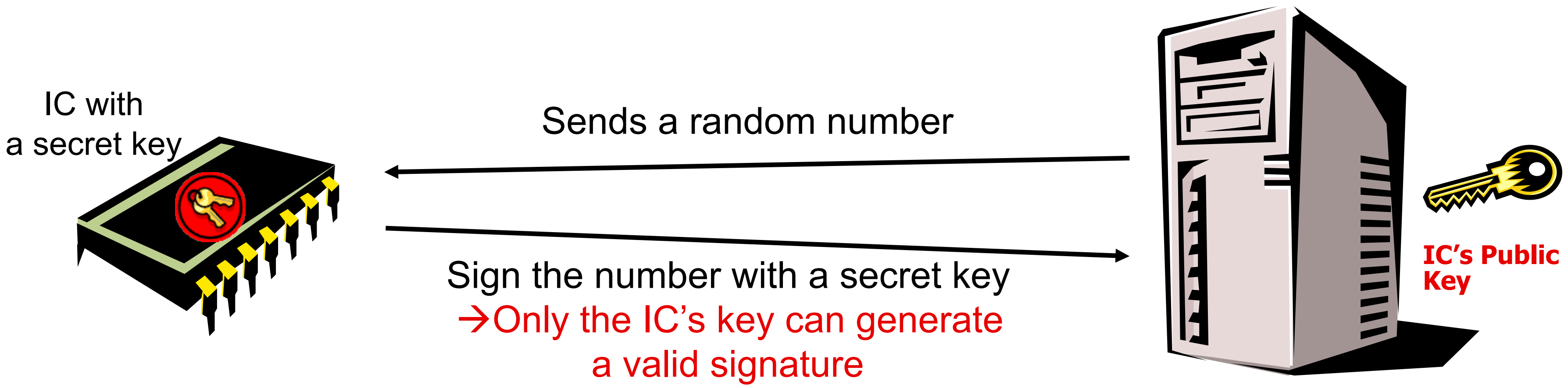
Physical Unclonable Functions (PUFs)

Srini Devadas

Traditional Authentication

- Each IC needs to be unique
  - Embed a unique secret key SK in on-chip non-volatile memory

- Use cryptography to authenticate an IC
  - A verifier sends a randomly chosen number
  - An IC signs the number using its secret key so that the verifier can ensure that the IC possesses the secret key



Physical Unclonable Function

A Physical Unclonable Function (PUF) is a function that is:

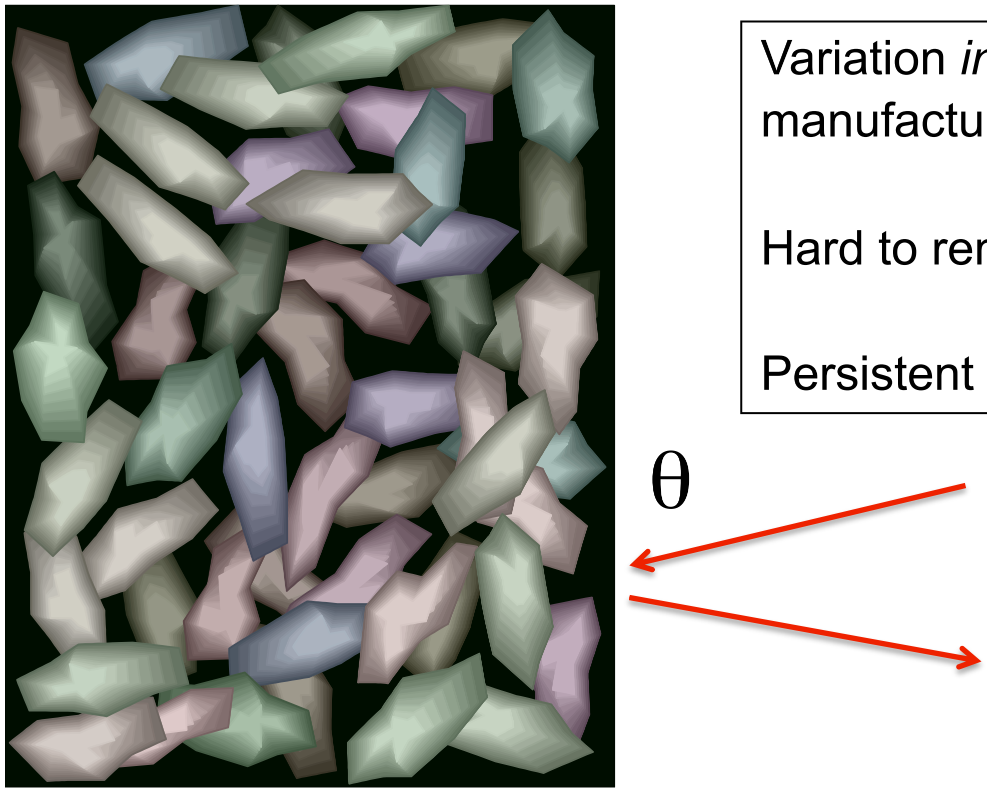
- Based on a physical system
- Easy to evaluate (using the physical system)
- Hard to predict

A PUF can additionally be:

- Manufacturer Resistant (better than unclonable: even the manufacturer cannot produce two identical systems)

Optical Physical Unclonable Functions

- **Generate secrets from a complex physical system**

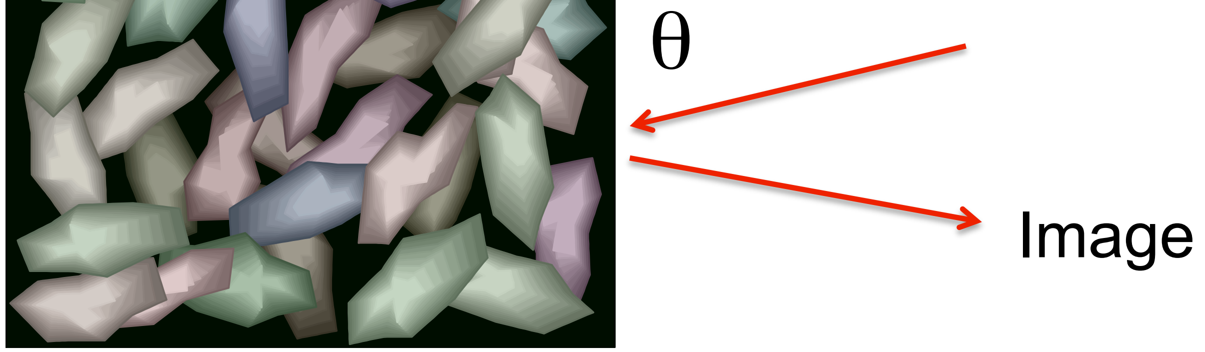


Variation *inherent* in (natural) manufacturing process

Hard to remove and predict

Persistent

Optical PUFs

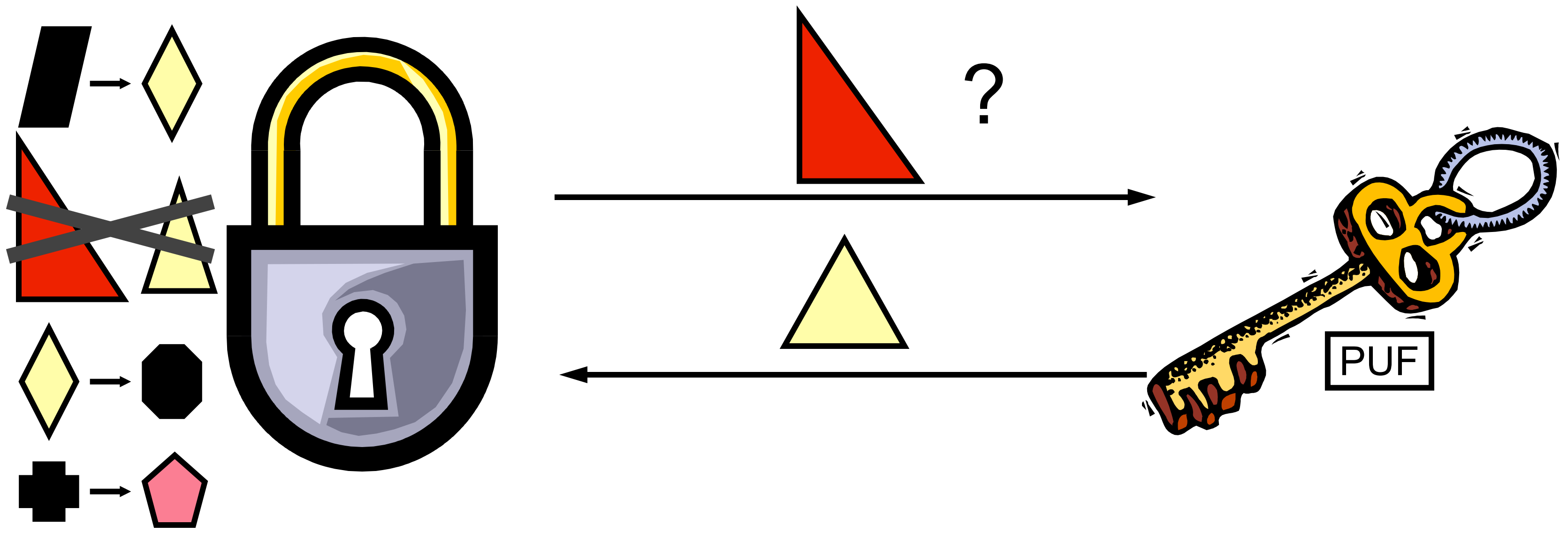


<u>Database</u>

θ_1	Image_1
θ_2	Image_2
θ_3	Image_3

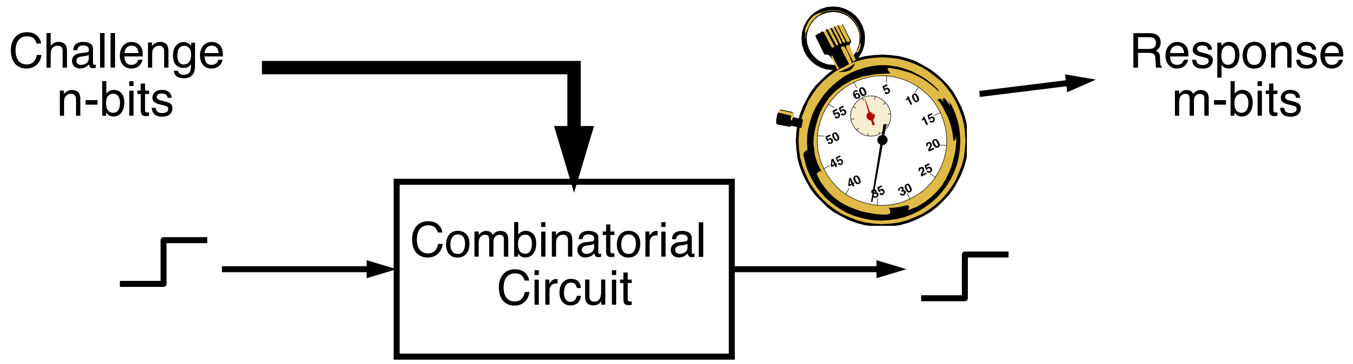
Using a PUF as a Key

- **A PUF can be used as a key.**
- **The lock has a secret database of challenge-response pairs.**
- **To open the lock, the key has to show that it knows the response to a challenge.**

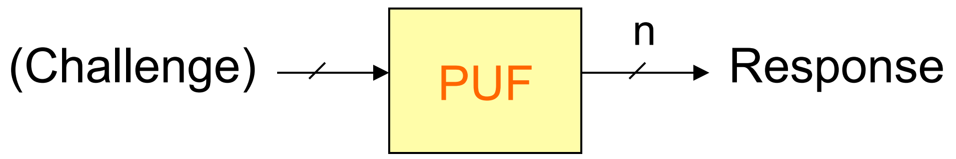


Silicon PUFs

- Because of random process variations, no two Integrated Circuits even with the same layouts are identical
  - Variation is inherent in fabrication process
  - Hard to remove or predict
  - Relative variation increases as the fabrication process advances
- Delay-Based Silicon PUF concept (2002)
  - Generate secret keys from unique delay characteristics of each processor chip♪

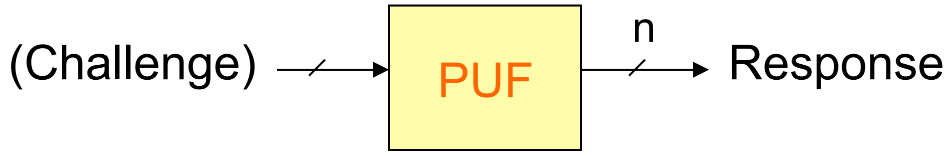


Why PUFs?



- PUF can enable secure, low-cost authentication w/o crypto
  - Use PUF as a function: challenge → response
  - Only an authentic IC can produce a correct response for a challenge
  - Inexpensive: no special fabrication technique
- PUF can generate a unique secret key / ID
  - Highly secure: volatile secrets, no need for trusted programming
  - Can integrate key generation into a secure processor
- Physical security: PUF secrets are the delays of wires and gates which are harder to extract via microscopy than bits in non-volatile memory

Main Questions



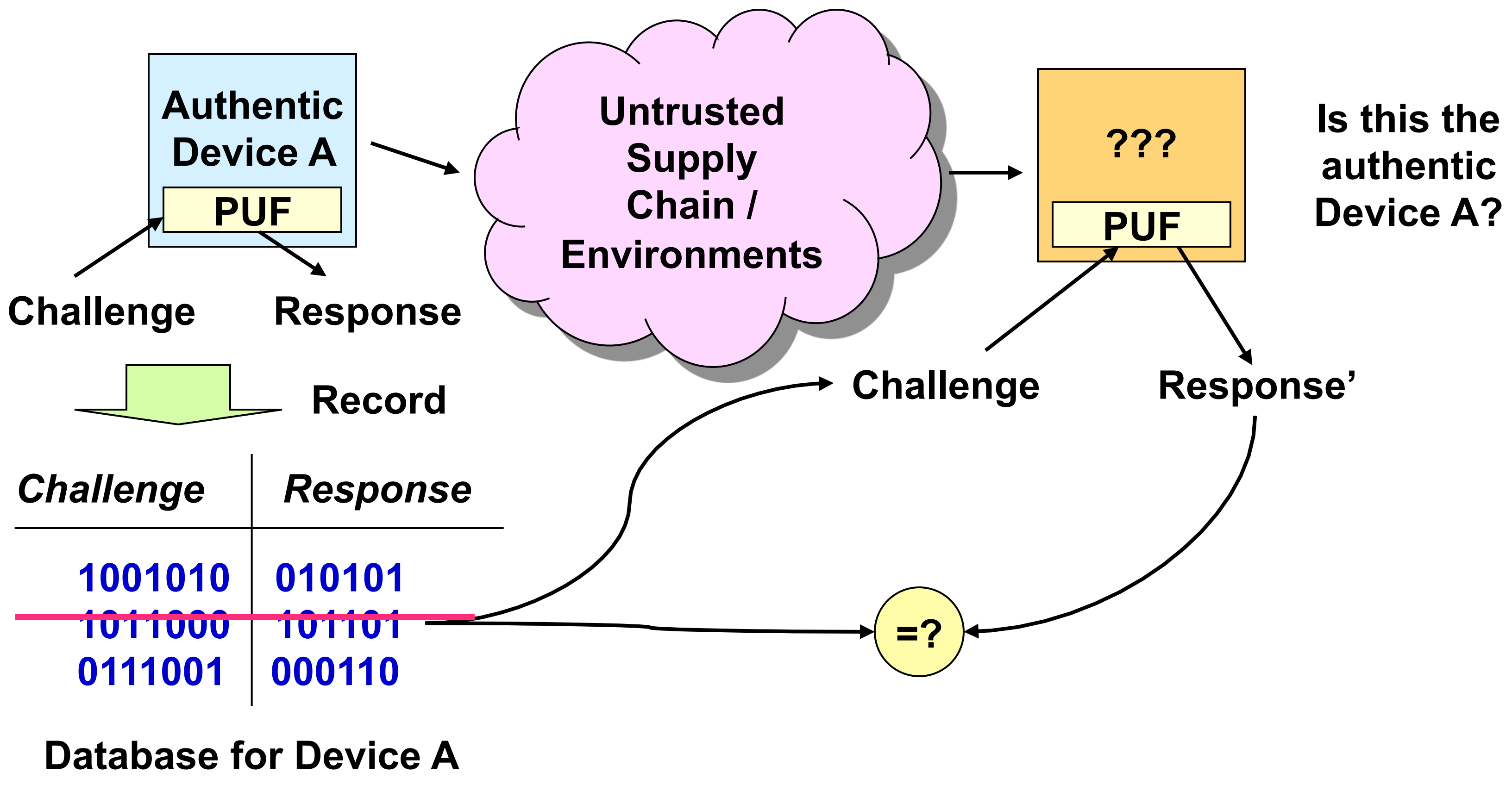
- How to design a PUF circuit for reliability and security?
  - Analog or asynchronous systems are susceptible to noise
  - Need barriers against software modeling attacks (equivalent to cryptanalysis)

- How to use the PUF for authentication and key generation?

Authentication Using PUFs

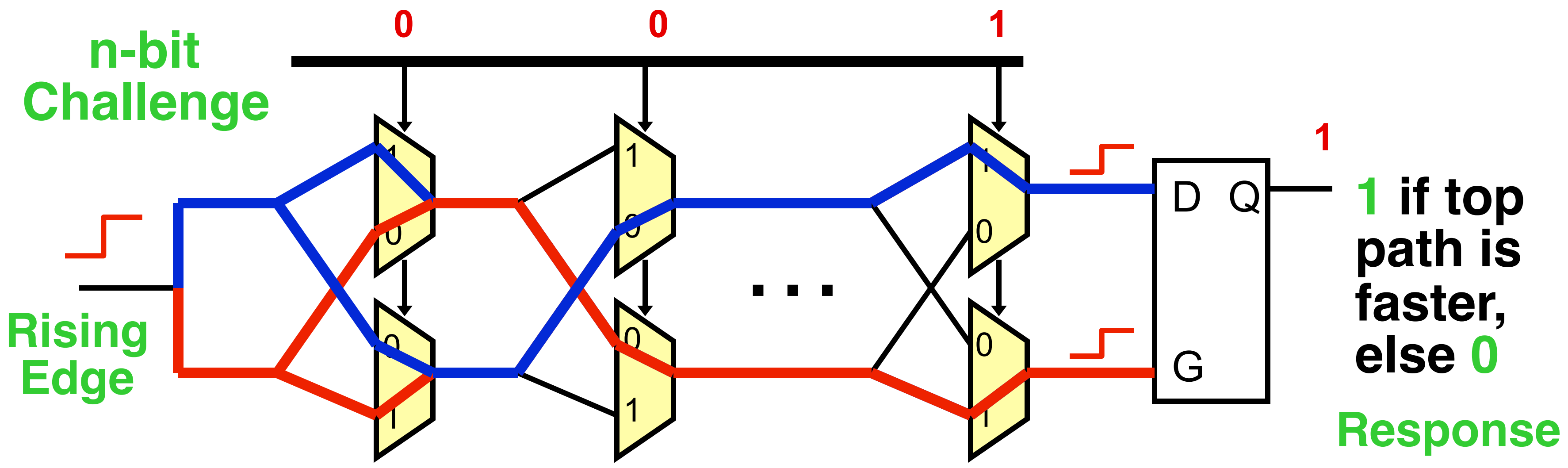
Low-Cost Authentication

- Protect against IC/FPGA substitution and counterfeits without using cryptographic operations



Challenge	*Response*
1001010	010101
~~1011000~~	~~101101~~
0111001	000110

An Arbiter-Based Silicon PUF



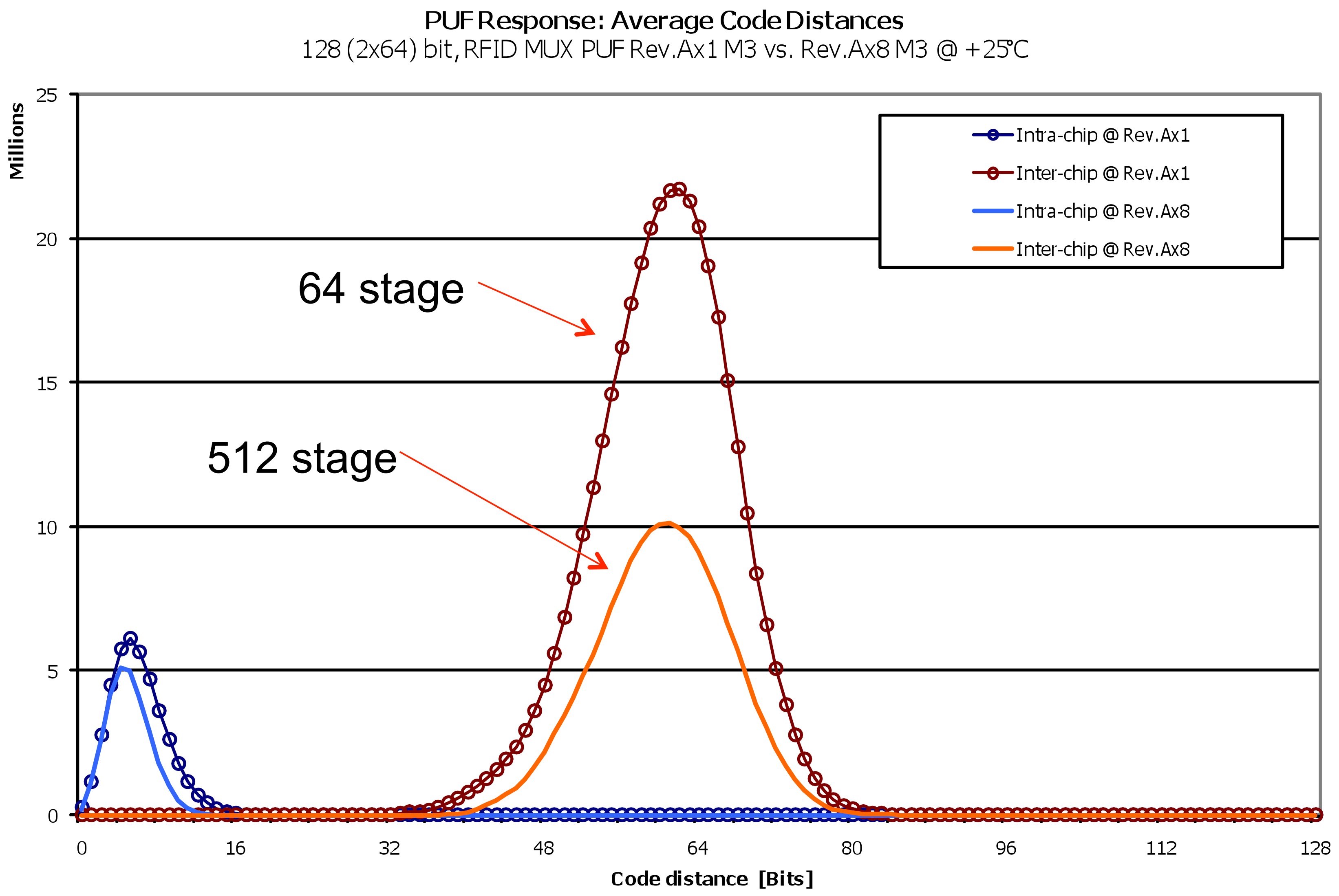
- Compare two paths with an identical delay in design
  - Random process variation determines which path is faster
  - An arbiter outputs 1-bit digital response
- Multiple bits can be obtained by either duplicate the circuit or use different challenges
  - Each challenge selects a unique pair of delay paths

Metrics

- Security: Show that different PUFs (ICs) generate different bits
  - ***Inter-chip*** variation: how many PUF bits (in %) are different between PUF A and PUF B?
  - Ideally, inter-chip variation should be close to 50%

- Reliability: Show that a given PUF (IC) can re-generate the same bits consistently
  - ***Intra-chip*** variation: how many bits flip when re-generated again from a single PUF
  - Environments (voltage, temperature, etc.) can change
  - Ideally, intra-chip variation should be 0%

Arbiter PUF Experiments: 64 and 512 stages

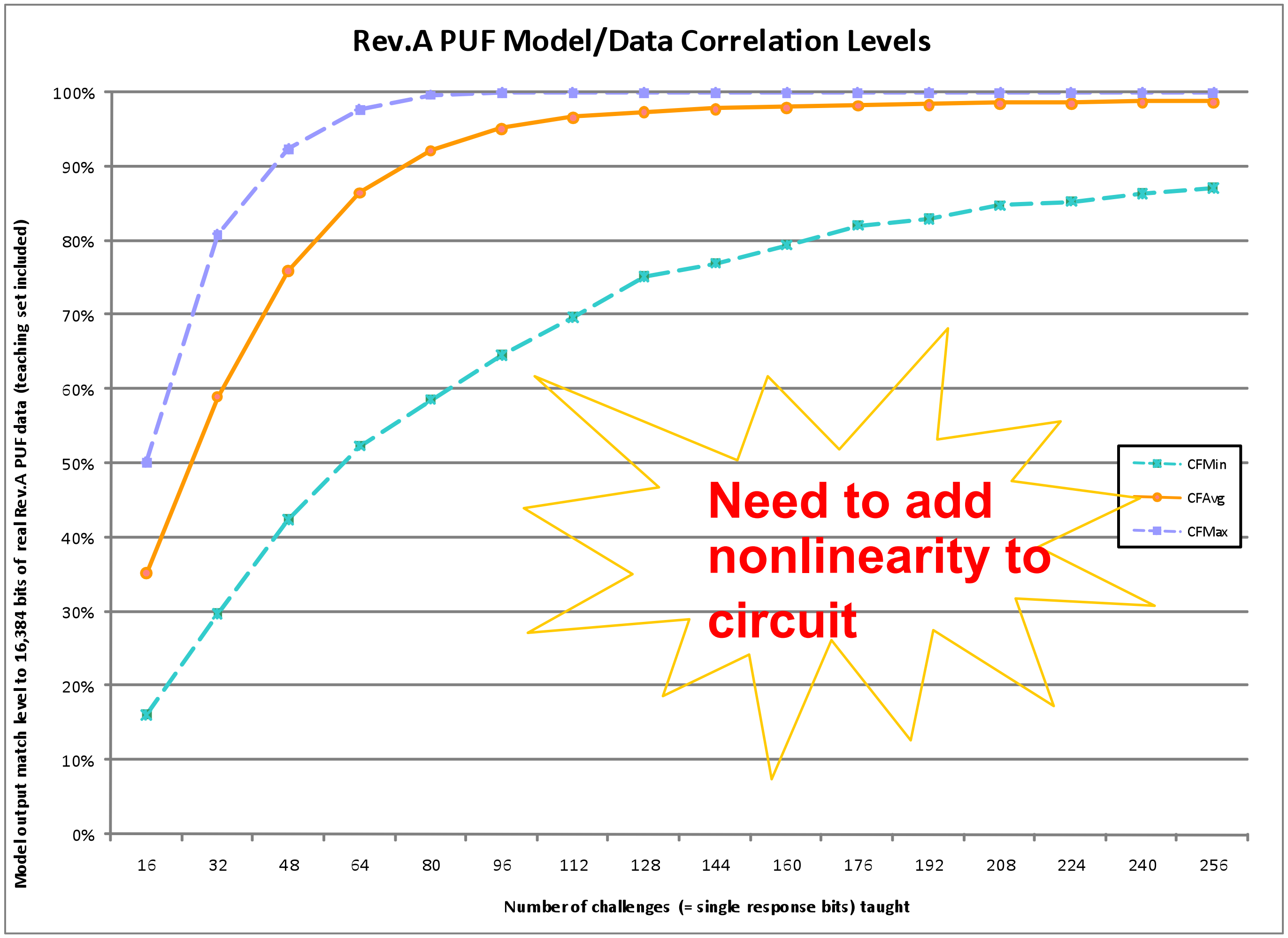


Attacking a PUF

- Duplication: Make more PUFs from the original blueprints and hope for a match.
- Brute-force attack: Exhaustively characterize the PUF by trying all challenges.
- Model building attack: Try to build a model of the PUF that has a significant probability of outputting the same value.
  - Discover hidden delays of wires/gates in a given PUF
- Direct measurement: Open the PUF and attempt to directly measure its characteristics.

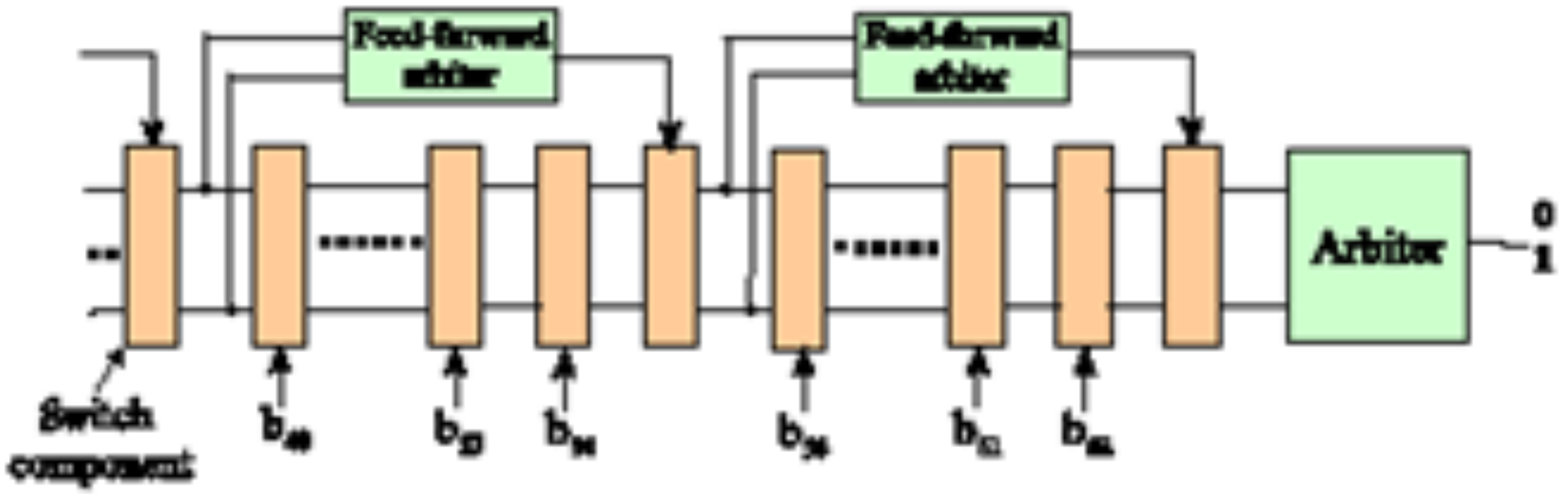
Arbiter PUF is not a PUF (clonable!)

- Introduced in 2003 paper, shown in same paper to be susceptible to a machine learning model-building attack



Feed-forward Arbiter

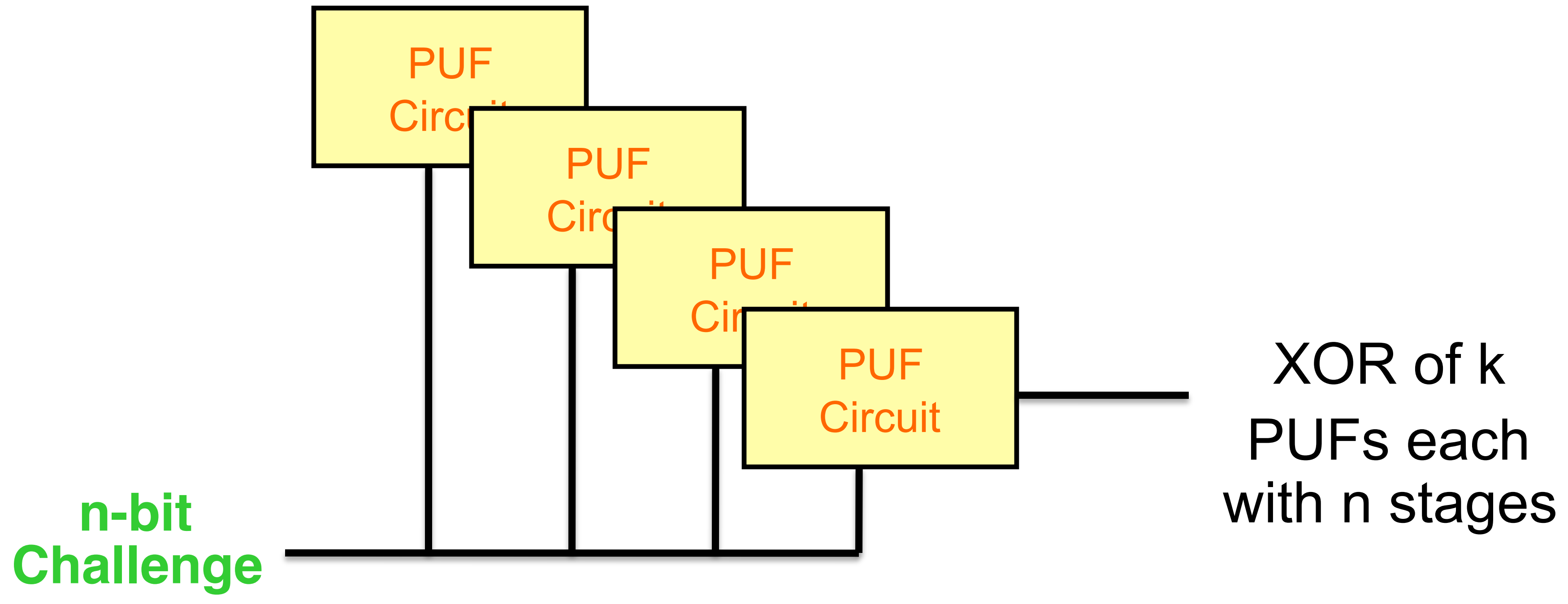
- Also introduced in 2003 paper, conjectured to be hard to learn



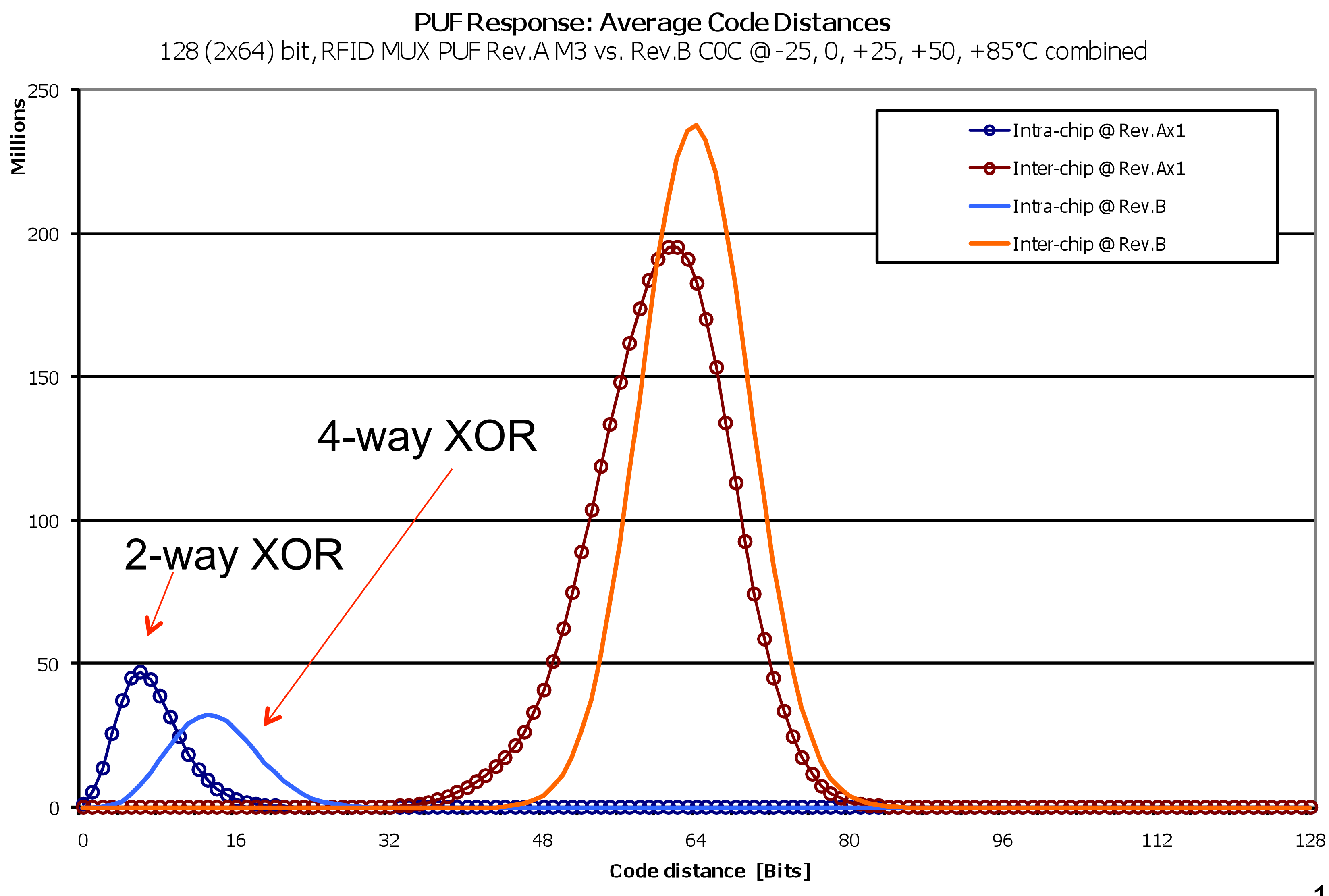
- Shown in 2008 (Koushanfar) and 2009 (Ruhrmair) to be susceptible to a model-building attack based on evolutionary algorithm

XOR Arbiter PUF

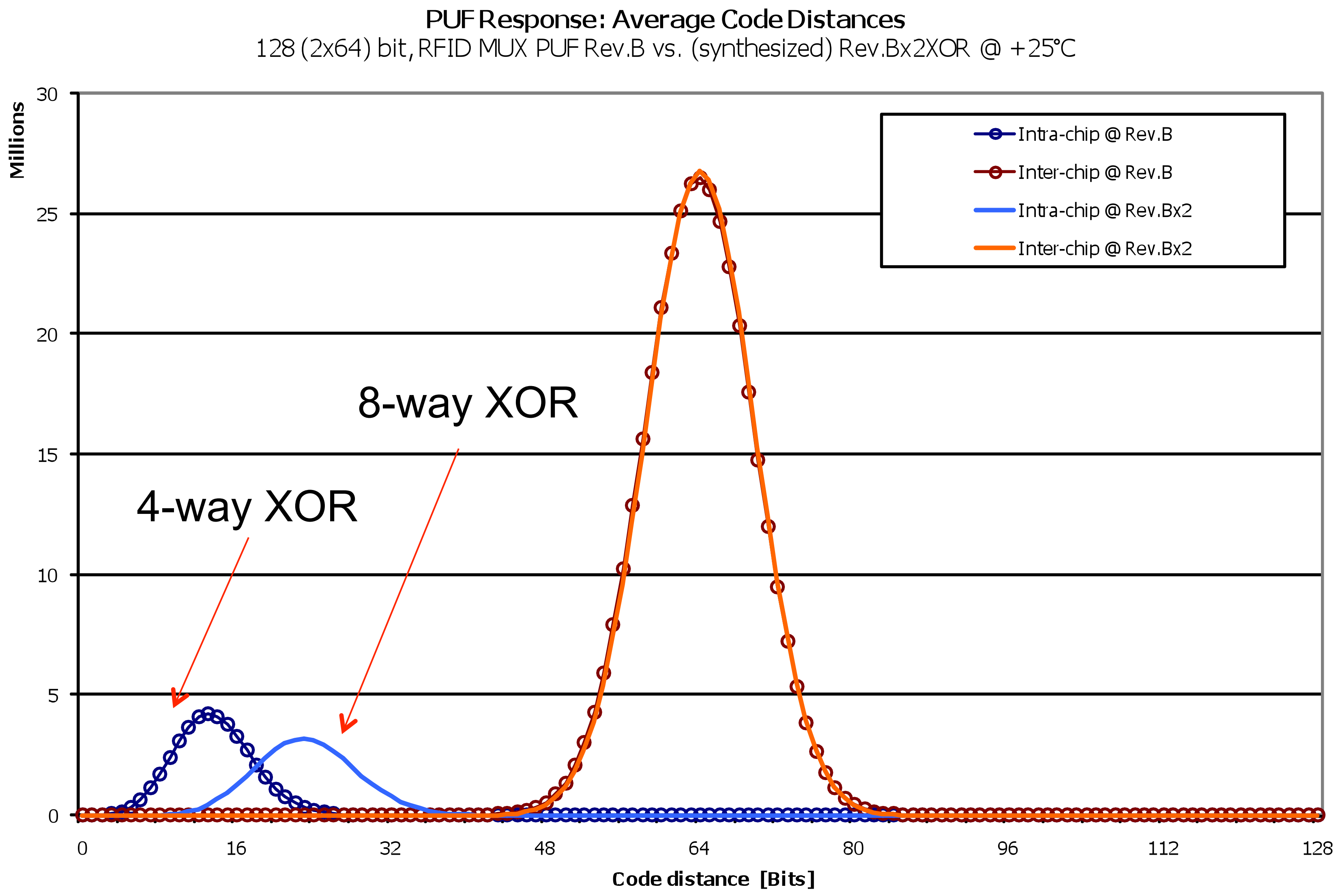
- Can process and combine outputs of multiple PUFs
- Simplest version: XOR operation



4-way XOR Experiments



8-way XOR experiments



XOR Arbiter PUF Security Analysis

- Logistic regression with Rprop heuristic is the best machine learning attack currently known on the XOR arbiter
- XOR arbiter is linearized by increasing the number of dimensions in the machine learning problem
  - Number of independent dimensions is ~ $n^k / k!$
- Machine learning runtime complexity grows as O (n^k) for k-way XOR over n-stage PUFs
  - Size of circuit grows as O(nk)

XOR Arbiter PUF Modeling Results

Ruhrmair et al, in submission

- $n = 64$, $k = 6$, and $n = 128$, $k = 5$ can be broken in a few days of CPU time on server farm
- Can go up to $k = 8$ with reasonable noise levels
- Increasing n does not increase noise and increases adversary’s computational effort
- Open questions:
  - Can we show a hardness result, i.e., learning requires time exponential in k?
  - Other ways of adding nonlinearity to circuit?

Potential Uses of PUFs

- Limited use transit token ticket

- Anti-counterfeiting applications

Next Time

Secure Processors