UNIVERSITÀ DEGLI STUDI DI SALERNO
DIPARTIMENTO DI INFORMATICA

Lecture 8 – Physically Unclonable Functions

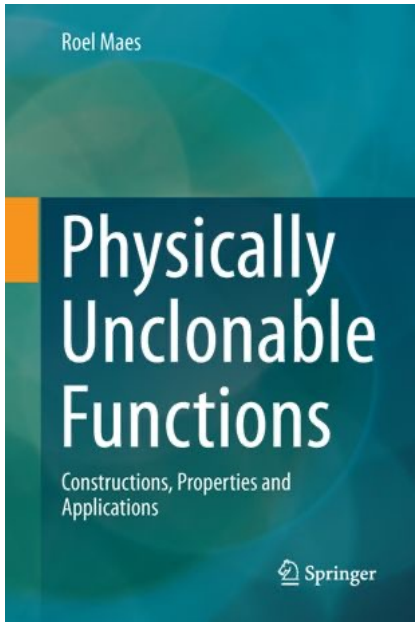
Prof. Esposito Christian

::.. Summary

- Physically Unclonable Functions
  - Introduction and Key Concepts
  - Architectures and Solutions
  - Applications

::.. References

- Roel Maes, “Physically Unclonable Functions”, Springer-Verlag Berlin Heidelberg 2013.

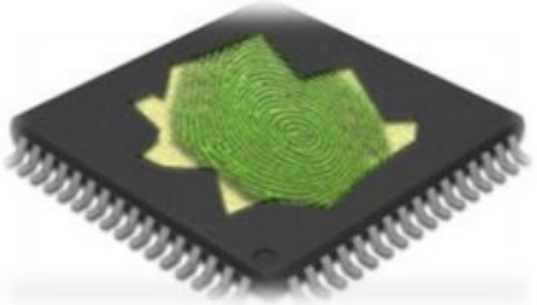


::.. Key Lectures

- Ruhrmair, Ulrich, and Jan Solter. "PUF modeling attacks: An introduction and overview." 2014 Design, Automation & Test in Europe Conference & Exhibition (DATE). 2014.
- Neshenko, Nataliia, et al., "Demystifying IoT security: an exhaustive survey on IoT vulnerabilities and a first empirical look on Internet-scale IoT exploitations", IEEE Communications Surveys & Tutorials, vol. 21, no. 3, pp. 2702-2733, thirdquarter 2019.
- Mosenia, Arsalan, and Niraj K. Jha, "A comprehensive study of security of Internet-of-Things", IEEE Transactions on Emerging Topics in Computing, vol. 5, no. 4, pp. 586-602, Ott.-Dic. 2016.

Introduction to PUFs

::.. PUF (1/12)



Using intrinsic random physical features to identity objects, systems and people is not new. Fingerprint identification is the key mechanism for human authentication. **Physical(ly) Unclonable Functions** (PUF) aims at realizing something similar to biometrics to IC by exploiting intrinsic electronic features of circuits.

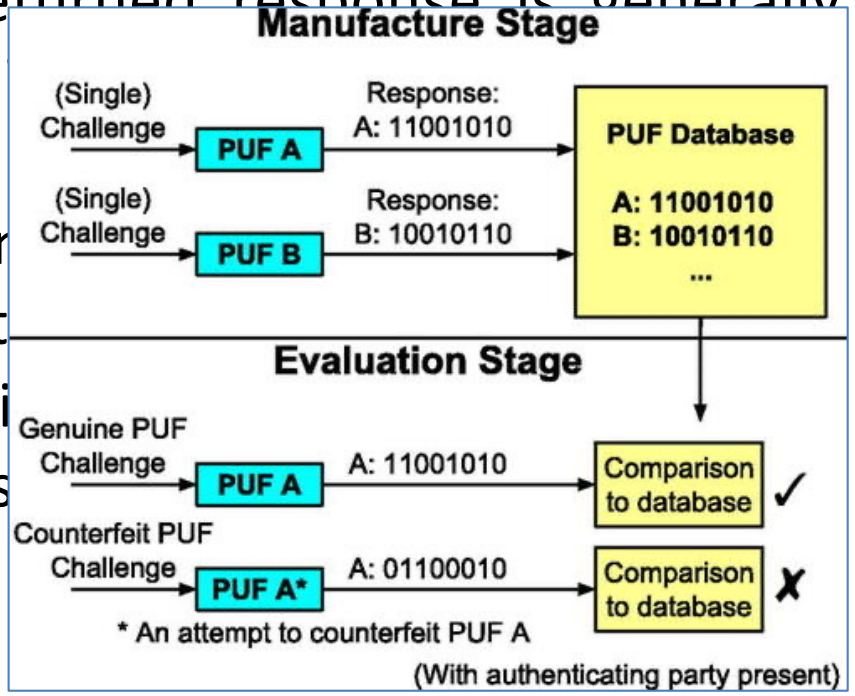
PUF realizes a functional operation, i.e., when provided with a certain input, or challenge, it returns a given response. PUF is not a true function in the mathematical view, as multiple possible outputs corresponds to a given input.

::.. PUF (2/12)

An applied challenge and the returned response is generally called as challenge-response pair or CRP, and the relation enforced between challenges and responses by a specific PUF is called CRP behavior. The response is considered a random variable, and the distribution of its PUF responses is needed to be known. CRP behavior is implied by the PUF construction, while for others it is less obvious, and some settings must be explicitly indicated.

::.. PUF (2/12)

An applied challenge and the returned response is generally called as challenge-response pa
enforced between challenges and
called CRP behavior. The respor
variable, and the distribution of it
be known. CRP behavior is impli
while for others it is less obvious
explicitly indicated.



PUF is used in two distinct phases:

- Enrollment Phase – a number of CRP is collected from a given PUF and stored in a CRP database
- Verification phase – a randomly-selected challenge is applied to the PUF and the obtained response is compared with the corresponding response in the database.

::.. PUF (3/12)

- A PUF response intra-distance is a random variable describing the distance between two responses from the same PUF instance when using the same challenge. It expresses the notion of noise by measuring the average reproducibility of a measured response with respect to an earlier observation of the same response.
- A PUF response inter-distance is a random variable describing the distance between two responses from different PUF instances when using the same challenge. It expresses the notion of uniqueness by measuring the average distinguishability of two PUF-based systems.

If the responses are string of bits, the Hamming distance is used. The intra-distance should be extremely small, while the inter-distance should have a Hamming distance of 50%.

::.. PUF (4/12)

Originally, PUF stood for physical unclonable function, but later the variant physically unclonable function also got into use. Despite being still used unchangeably, there is a small difference in meaning.

1. A physical unclonable function is ‘a physical function which is unclonable’;
2. A physically unclonable function is ‘a function which is physically unclonable’.

The second interpretation is a more fitting description of the actual concept of a PUF.

The adjective unclonable is the central specifier in the acronym, reflecting the distinguishing property of a PUF. However, the actual meaning of unclonable is not clearly defined.

::.. PUF (5/12)

It is crucial to define what a clone is:

1. Mathematical clone - any construction that accurately mimics the CRP behavior of a particular PUF could be considered a clone thereof.
2. Physical clone - any construction that has the same CRP behavior and the same physical appearance as the a particular PUF, or even manufactured in the same production process, is a clone.

PUF should be made unclonable with respect to the mathematical rather than the physical definition of a clone.

A PUF is not even strictly a function, since a single challenge can be related to more than a single response due to the uncontrollable effects of the physical environment and random noise on the response generation.

::.. PUF (6/12)

A more fitting mathematical description of a PUF is as a probabilistic function, i.e., a function for which part of the input is an uncontrollable random variable.

PUFs and PUF-like constructions have been proposed based on a wide variety of technologies and materials such as glass, plastic, paper, electronic components and silicon integrated circuits (ICs). A possible classification of PUF constructions is based on the electronic nature of their identifying features.

A PUF construction is defined intrinsic PUFs, if it meets at least two conditions:

1. its evaluations are performed internally by embedded measurement equipment,
2. its random instance-specific features are implicitly introduced during its production process.

::.. PUF (7/12)

1. An external evaluation consists in the measurement of features which are externally observable and/or which is carried out using equipment which is external to the physical entity containing the features.
2. Internal evaluations implies measurements of internal features which are carried out by equipment completely embedded in the instance itself.

There are several advantages to employ internal measurements/evaluations:

1. Every PUF instance can evaluate itself without any external limitations or restrictions. Internal evaluations are typically also more accurate since there is less opportunity for outside influences and measurement errors.

::.. PUF (8/12)

2. As long as an instance does not disclose a response to the outside world, it can be considered an internal secret. This is of particular interest when the embedding object is a digital IC, since in that case the PUF response can be used immediately as a secret input for an embedded implementation of a cryptographic algorithm.

A possible disadvantage of an internal evaluation is that one needs to trust the embedded measurement equipment, since it is impossible to externally verify whether the measurement takes place as expected.

A second construction-based distinction considers the source of the randomness of the evaluated features in a PUF.

::.. PUF (9/12)

1. An explicit randomization procedure can be present within the manufacturing process of the PUF instances with the sole purpose of introducing random features which will later be measured when the PUF is evaluated.
2. The measured random features arise naturally as an artifact of uncontrollable implicit side effects during the manufacturing process.

There are some subtle but interesting advantages :

1. Implicit random variations generally come at no extra cost since they are already (unavoidably) present.
2. Implicit random variability in manufacturing is generally undesirable as manufacturers tries to reduce it as much as possible. However, completely avoiding it is technically impossible.

::.. PUF (10/12)

As a consequence, PUF constructions based on implicit process variations have the interesting security advantage that even the manufacturers cannot remove or control the random features at the core of the PUF's functionality.

A last classification is explicitly based on the security properties of their challenge-response behavior.

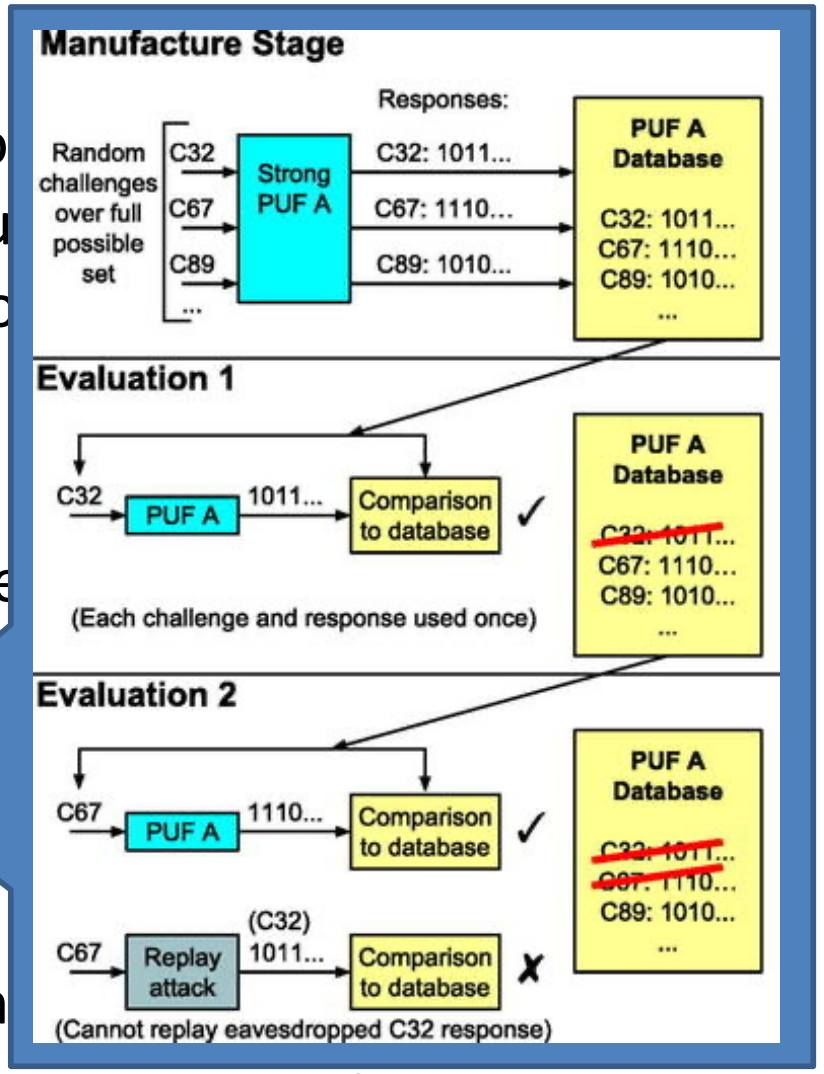
1. A PUF is called a strong if, even after giving an adversary access to a PUF instance for a prolonged period of time, it is still possible to come up with a challenge to which with high probability the adversary does not know the response.
2. PUFs which do not meet the requirements for a strong PUF, are consequentially called weak PUFs.

::.. PUF (10/12)

As a consequence, PUF constructio
variations have the interesting secu
manufacturers cannot remove or co
the core of the PUF's functionality.

A last classification is explicitly base
of their challenge-response behavi

1. A PUF is called a strong
access to a PUF instance for a
still possible to come up with a
probability the adversary does n
2. PUFs which do not meet the requirements for a strong PUF, are consequentially called weak PUFs.



::.. PUF (11/12)

A strong PUF has

- a very large challenge set, since otherwise the adversary can simply query all challenges and no unknown challenges are left,
- the capability of making infeasible to built an accurate model of its observed CRP, or in other words exhibiting an unpredictable CRP behavior.

An extreme case called a Physically Obfuscated Key (POK) is a PUF construction which has only a single challenge, and represents a means to store keys inside an IC.

Constructing a practical (intrinsic) strong PUF with strong security guarantees turns out to be very difficult, up to the point where one can consider it an open problem whether this is actually possible.

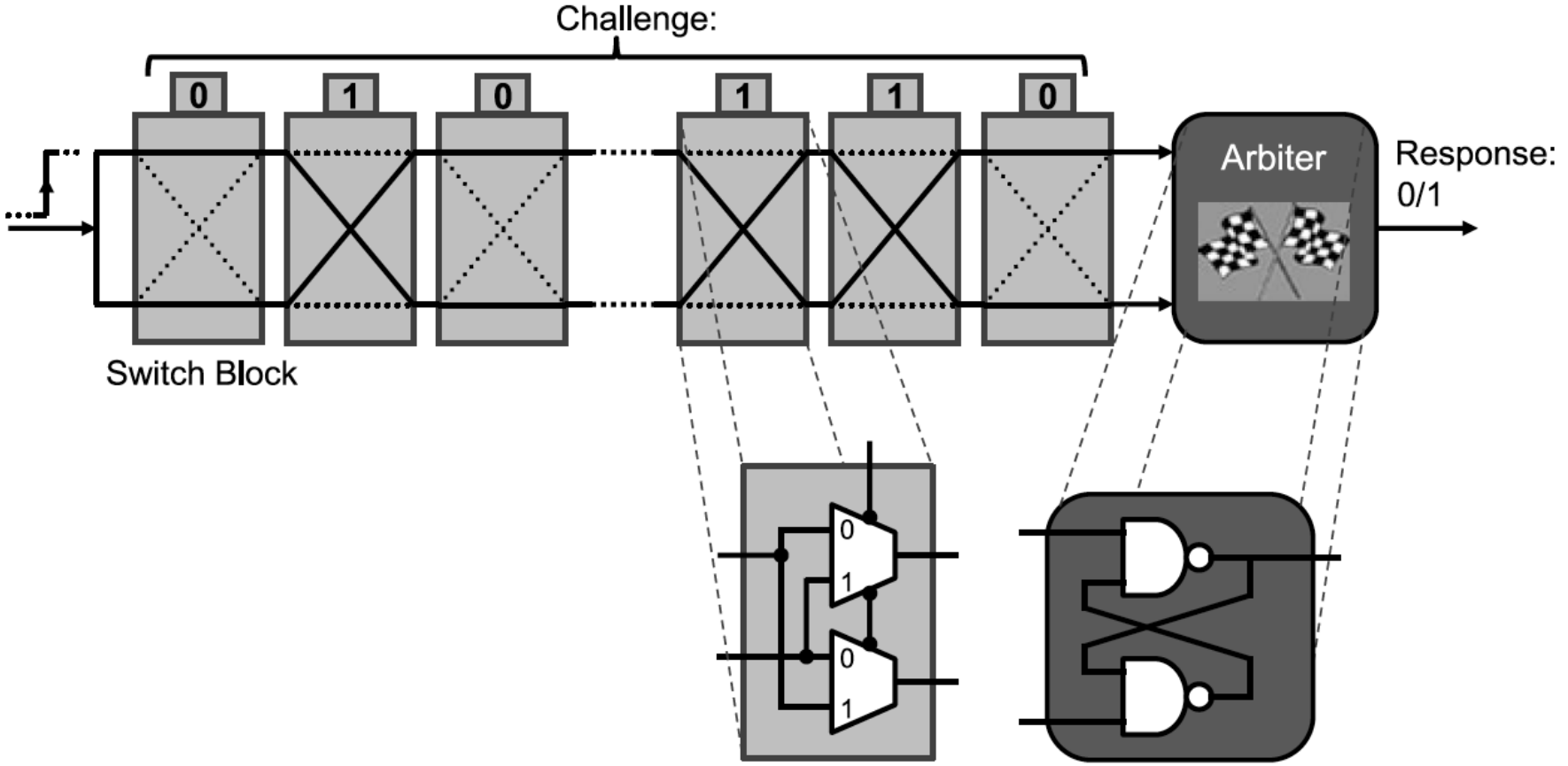
::.. PUF (12/12)

All known intrinsic PUF constructions are silicon PUFs based on random process variations occurring during the manufacturing process of silicon chips.

- Delay-based silicon PUFs – They measures random variations on the delay of a digital circuit;
- Memory-based silicon PUFs – They use random parameter variations between matched silicon devices, also called device mismatch, in bistable memory elements;
- Intrinsic silicon PUF - They consist of mixed-signal circuits where an embedded analog measurement is quantized with an analog-to-digital conversion to produce a digital response representation.

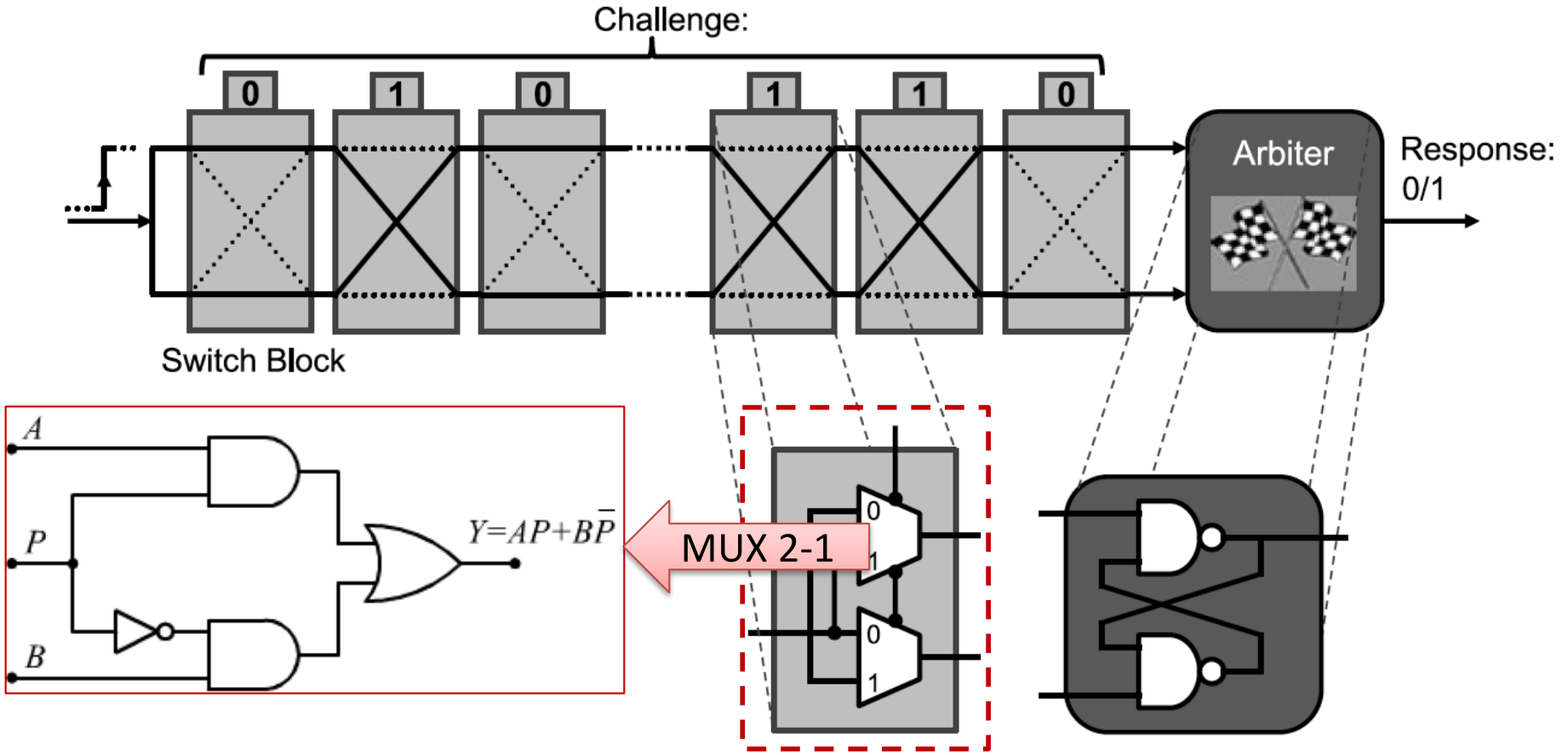
::.. Arbiter PUF (1/4)

The arbiter PUF is a type of delay-based silicon PUF and the driving idea is to explicitly introduce a race condition between two digital paths on a silicon chip.



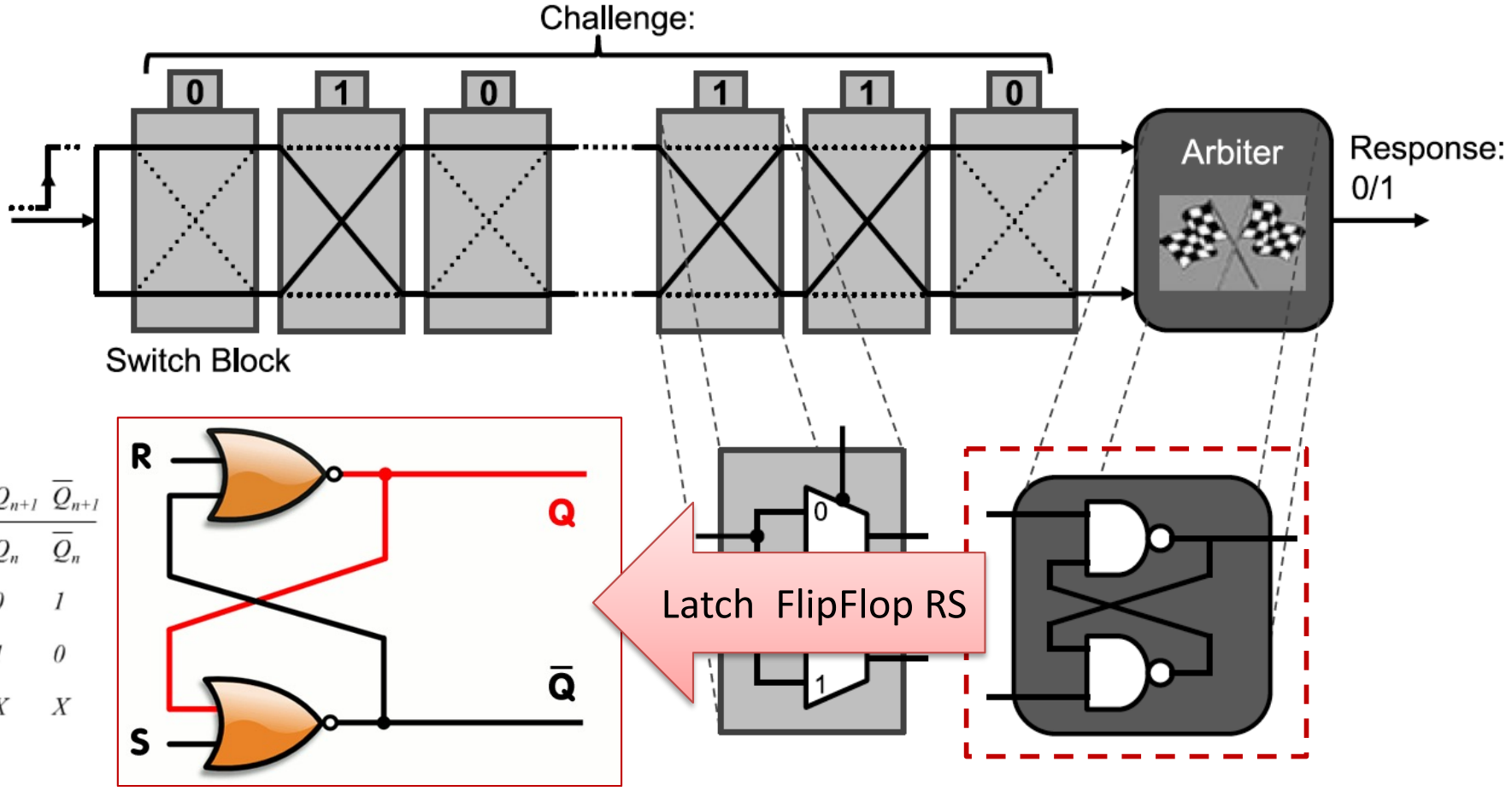
::.. Arbiter PUF (1/4)

The arbiter PUF is a type of delay-based silicon PUF and the driving idea is to explicitly introduce a race condition between two digital paths on a silicon chip.



::.. Arbiter PUF (1/4)

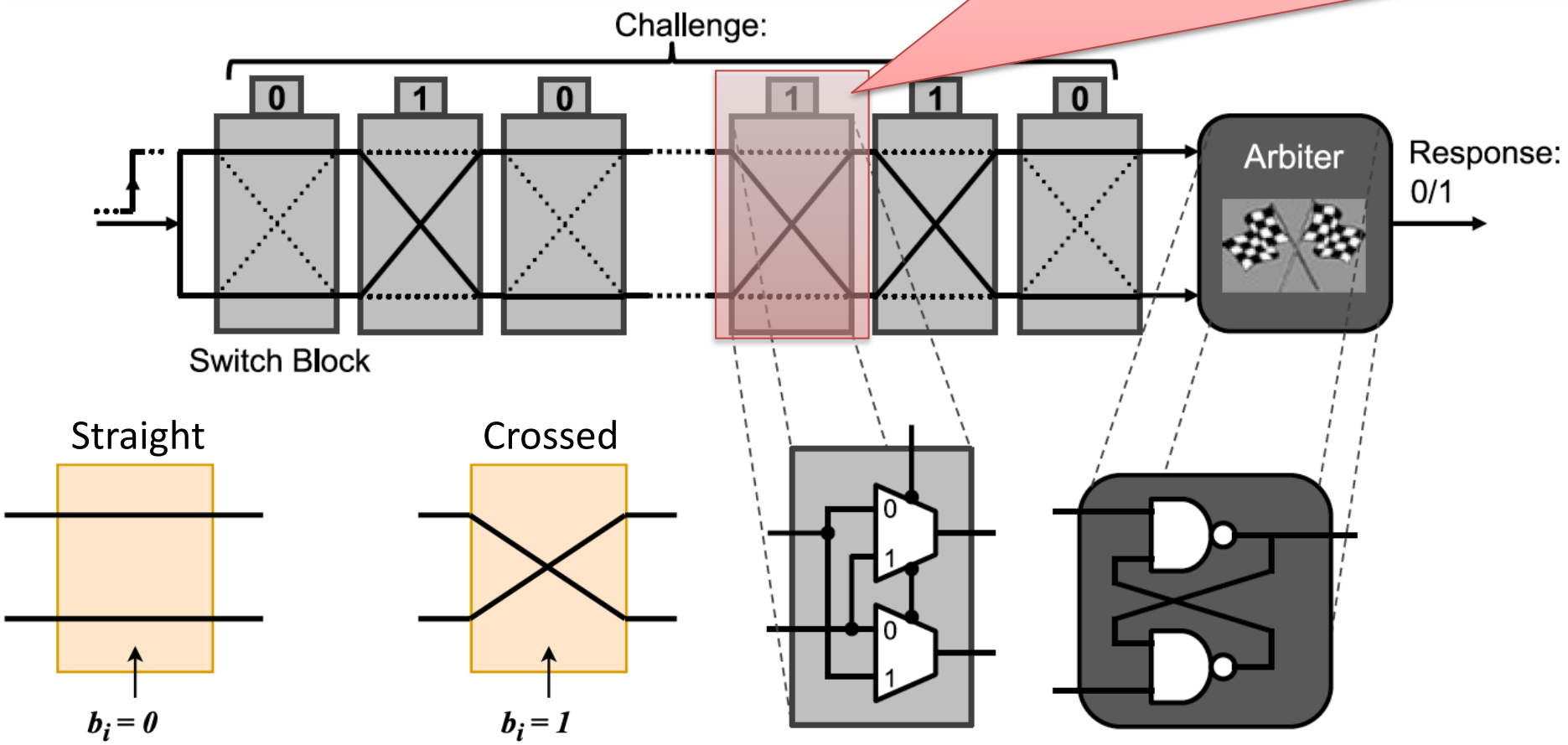
The arbiter PUF is a type of delay-based silicon PUF and the driving idea is to explicitly introduce a race condition between two digital paths on a silicon chip.



S	R	Q_{n+1}	$\overline{Q}_{n+1}$
0	0	Q_n	$\overline{Q}_n$
0	1	0	1
1	0	1	0
1	1	X	X

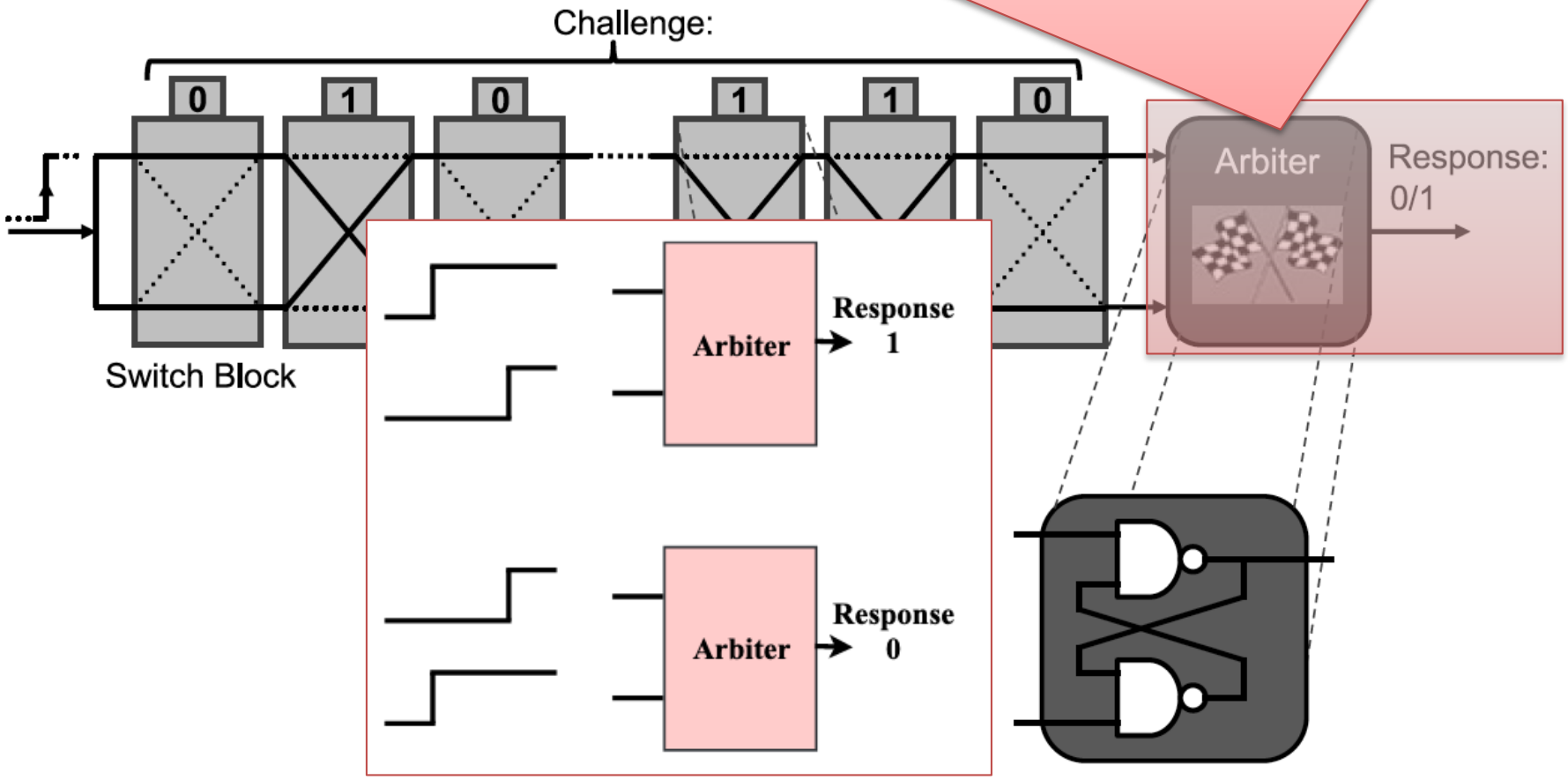
::.. Arbiter PUF (1/4)

Each switch block, consisting of two 2-to-1 multiplexers, can propagate an input signal through two configurable path, i.e., connected straight or switched. A configuration bit establishes in which configuration of the block has to work.



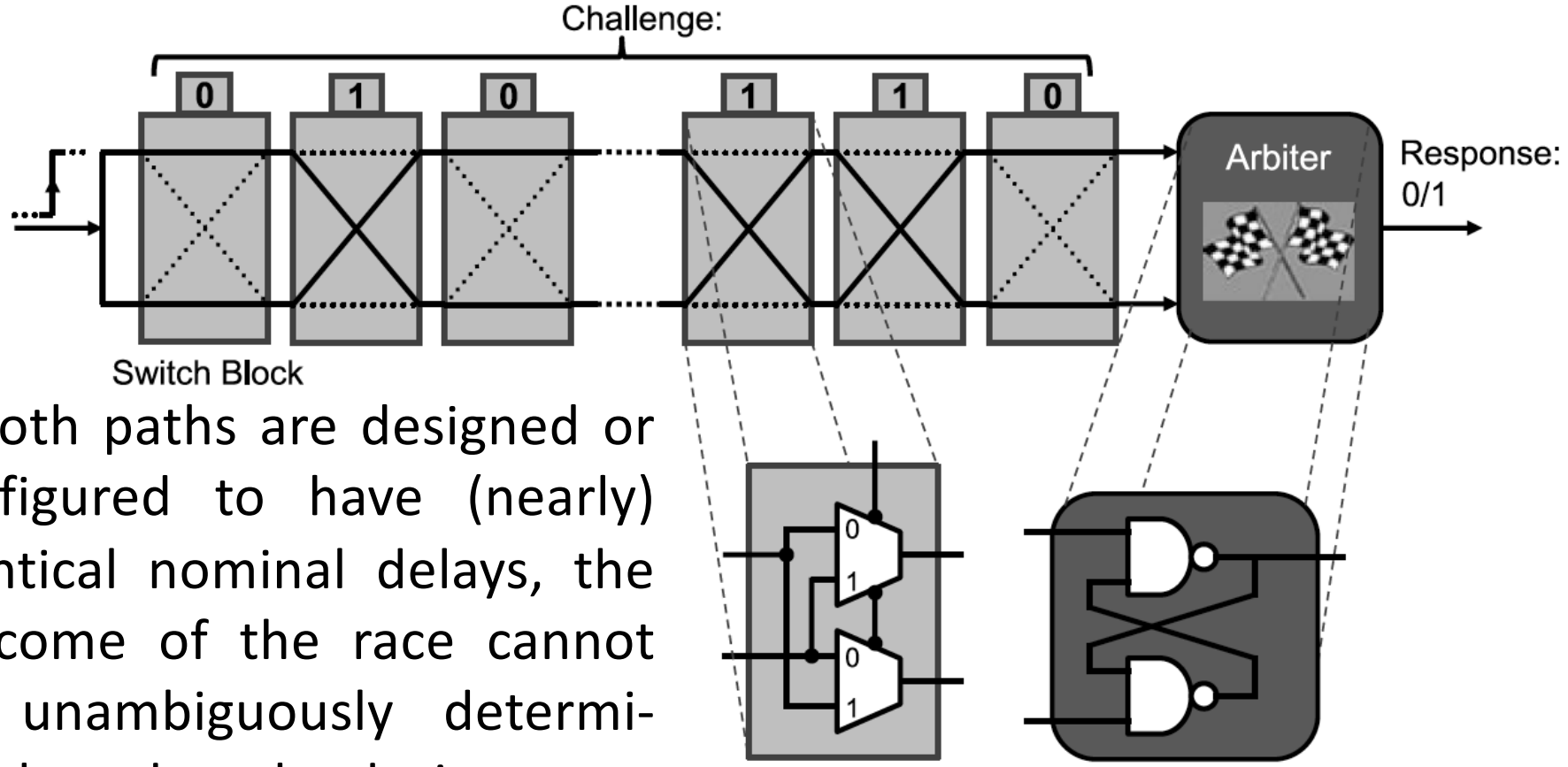
::.. Arbiter PUF (1/4)

At the end of the lines there is an arbiter circuit (made as an SR latch) able to resolve the race, i.e. it determines which of the two paths was faster and outputs a binary value accordingly.



::.. Arbiter PUF (1/4)

The arbiter PUF is a type of delay-based silicon PUF and the driving idea is to explicitly introduce a race condition between two digital paths on a silicon chip.



If both paths are designed or configured to have (nearly) identical nominal delays, the outcome of the race cannot be unambiguously determined based on the design.

::.. Arbiter PUF (2/4)

The actual delay experienced by an edge which travels the two paths will not be exactly equal, due to the effect of random silicon process variations. Since the effect of silicon process variations is random per device, but static for a given device, the delay difference and by consequence the arbiter output will be device specific.

There is a non-negligible possibility that by chance both delays are nearly identical. In that case, the two edges will reach the arbiter virtually at the same moment, causing the arbiter circuit to go into a metastable state, i.e., the logic output of the arbiter circuit is temporarily undetermined. After a short but random time, the arbiter leaves its metastable state and outputs a random value which is independent of the outcome of the race.

::.. Arbiter PUF (3/4)

These are the requirements for an Arbiter PUF of high quality:

1. The delay lines are designed to be nominally perfectly symmetrical, i.e., any difference in delay is solely caused by process variations.
2. The arbiter circuit is completely fair, i.e., it does not favor one of its inputs over the other. A basic SR latch is the best option since it is unbiased due to its symmetric construction.

If either of the two conditions is not met, the arbiter PUF is biased, which results in a lower uniqueness of its responses. Designing a perfectly unbiased arbiter PUF is highly non-trivial since it requires very low-level control over implementation choices. E.g., it has big bias when it is mapped on Field Programmable Gate Array (FPGA), as performing completely symmetric routing is physically infeasible.

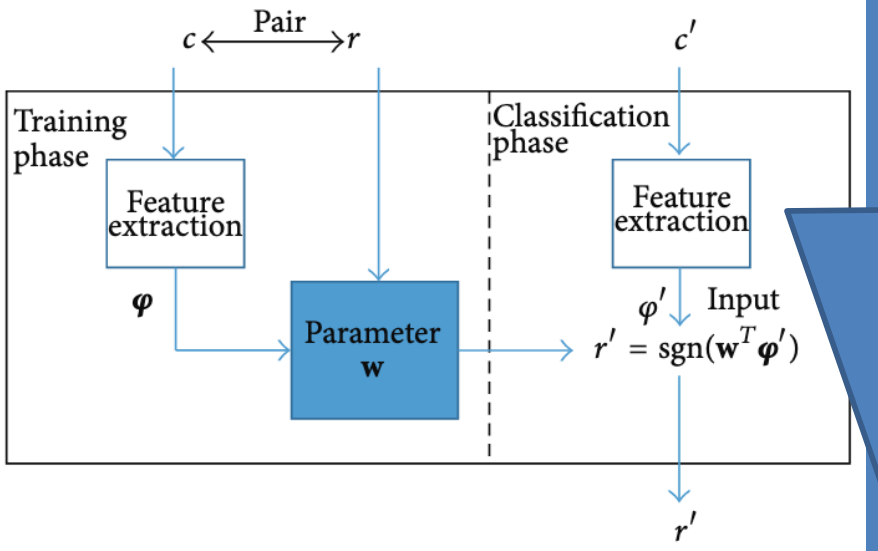
::.. Arbiter PUF (4/4)

The basic arbiter PUF construction with n chained switch blocks provides with 2^n different challenges. However, the number of delay parameters which determine the arbiter PUF response is only linear in n. In practice, when one learns the underlying delay parameters and one can model the interaction with the challenge bits, one is able to accurately predict response bits to random challenges, even without access to the PUF.

With a modeling attack, from the collected CRP it is possible to construct a model of the PUF, as a mathematical clone of the arbiter PUF.

A model can be obtained by using the sum of the delays of serialized components added to an estimation of unknown underlying delay parameters learned from observing CPR.

::.. Arbiter PUF (4/4)



During the training, several CRPs from the target Arbiter PUF are collected. Then, feature extraction is performed and a challenge c is transformed into φ to simplify the machine learning process. After, a model w is constructed.

A different challenge c', whose response is unknown, is provided during the classification phase. After feature extraction from c' to φ', the response r' is predicted by using the model w.

With a modeling attack, from

construct a model of the PU

arbiter PUF.

A model can be obtained b

serialized components adde

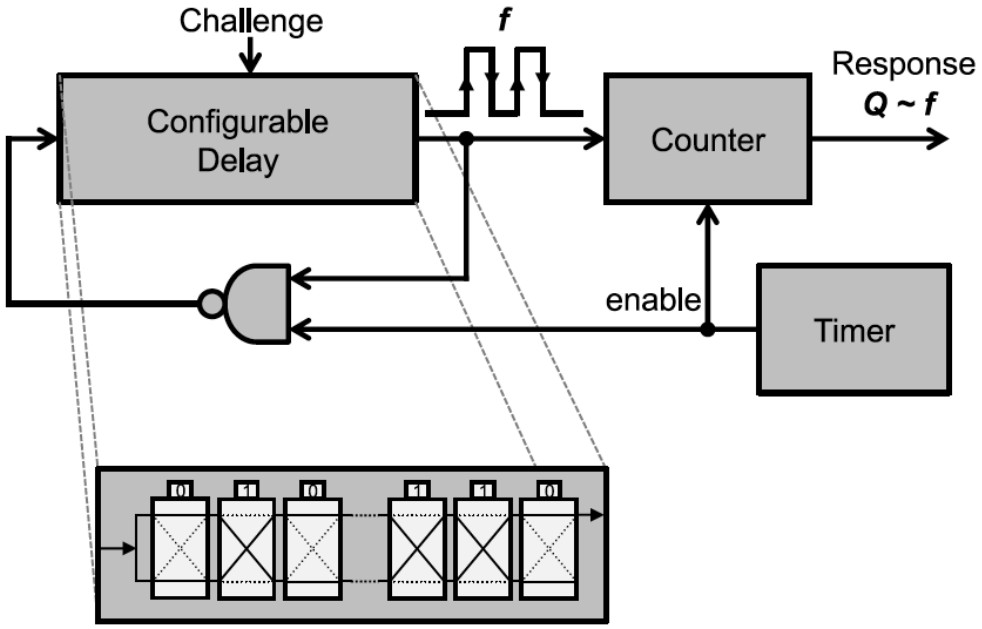
underlying delay parameters learned from observing CPR.

::.. Ring Oscillator PUF (1/4)

A second type of delay-based intrinsic PUF is the ring oscillator PUF, consisting in measuring random variations on the frequencies of digital oscillating circuits. Basically, the key element is a **ring oscillator**, i.e., a device composed of an odd number of NOT gates in a ring, whose output oscillates between two voltage levels. The uncontrollable effect of silicon process variations on the delay of digital components causes variation in the frequency of this oscillation.

A typical ring oscillator PUF construction has two basic components, ring oscillators and frequency counters, which are arranged in a particular architecture, and are often combined with a response-generating algorithm.

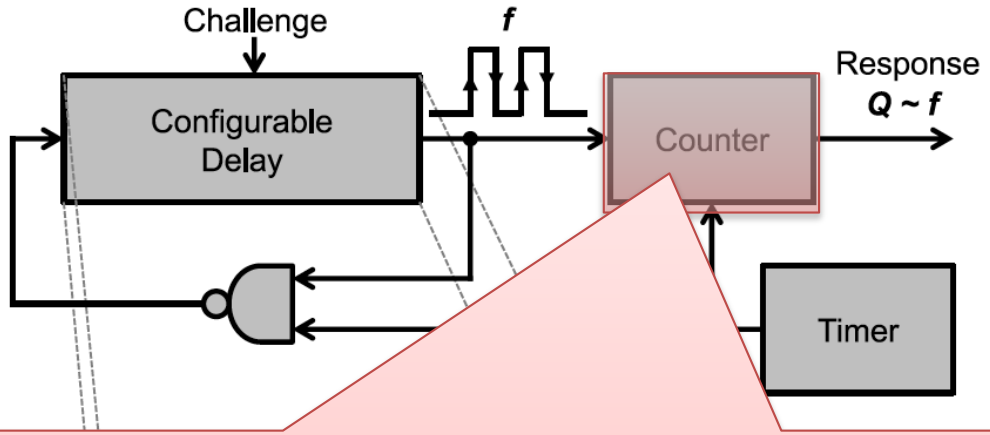
::.. Ring Oscillator PUF (2/4)



A very basic ring oscillator PUF is a variant of the switch block-based delay line. The delay circuit is transformed into an oscillator by applying negative feedback. An additional AND-gate in the loop allows us to en-able/disable the oscillation.

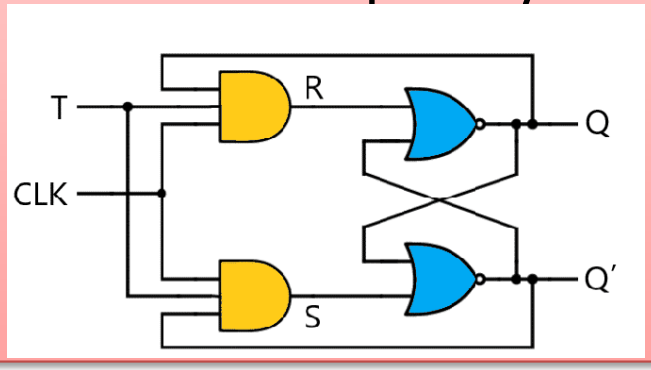
The oscillating signal is fed to a frequency counter, which counts the number of oscillating cycles in a fixed time interval. The resulting counter value is a direct measure of the loop's frequency.

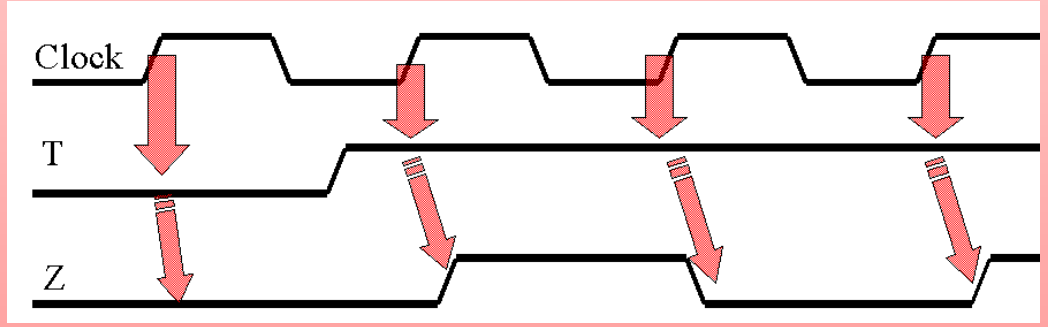
::.. Ring Oscillator PUF (2/4)



A very basic ring oscillator PUF is a variant of the switch block-based delay line. The delay circuit is transformed into an oscillator by applying

The oscillating signal is processed by a simple edge detector, such as an edge triggered T flip-flop, enabling the counter every time a rising edge is detected. This architecture is robust, but the edge detector limits the frequency of the ring oscillator to half the clock one.





::.. Ring Oscillator PUF (3/4)

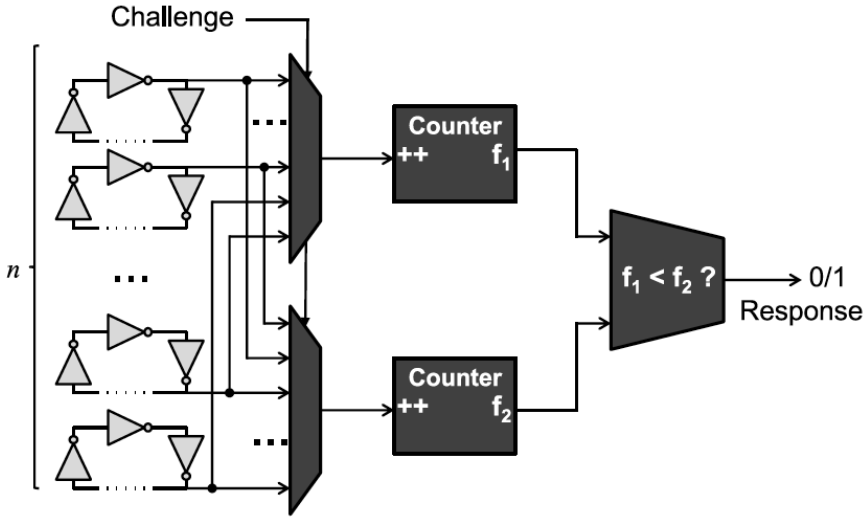
The influence of environmental conditions on the evaluation of this construction is significant, i.e. changes in temperature and voltage cause frequency changes which are orders of magnitude larger than those caused by process variations.

To counter this influence, a post-processing technique called **compensated measuring** has been proposed. It consists in evaluating the frequency of two ring oscillators on the same device simultaneously and consider a differential function.

Environmental changes will affect both frequencies in roughly the same way, and their ratio will be much more stable.

::.. Ring Oscillator PUF (4/4)

Since the ring oscillator is based on the same delay circuit as the simple arbiter PUF, this construction will also be susceptible to modeling attacks. Moreover, the produced response is an integer counter value, or a real value in the case of compensated measurement, and cannot be used directly as a bit string in subsequent building blocks. It first needs to be quantized in an appropriate way to obtain a bit string response.

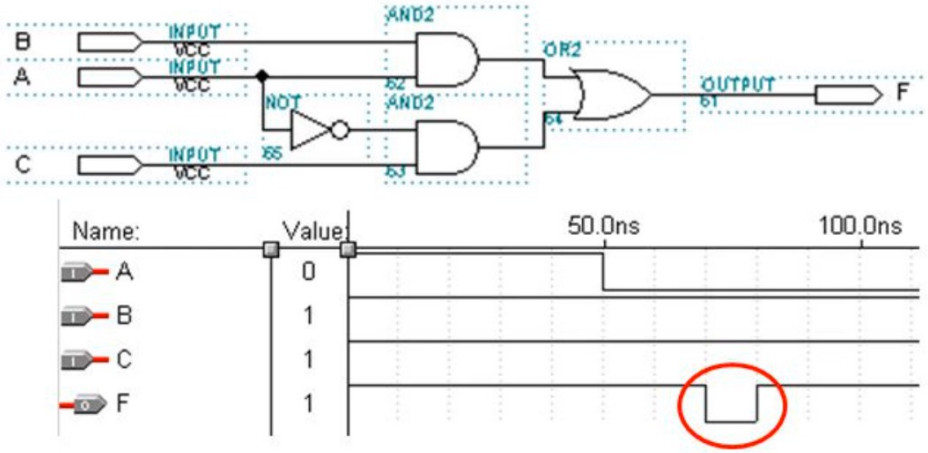


An alternative solution contains an array of n fixed but identical inverter chains with two frequency counters, which can be fed by each of the n inverter chains. Two n-to-1 muxes control which oscillators to be applied to the two counters.

::.. Glitch PUF (1/2)

A purely combinatorial circuit has no internal state: its steady-state output is entirely determined by its input signals. However, when the logical value of the input changes, transitional effects can occur, i.e. it can take some time before the output assumes its steady-state value. These effects are called glitches and the occurrence of glitches is determined by the differences in delay of the different logical paths from the inputs to an output signal.

- F = AB + A'C: assume all gate delays = 10ns



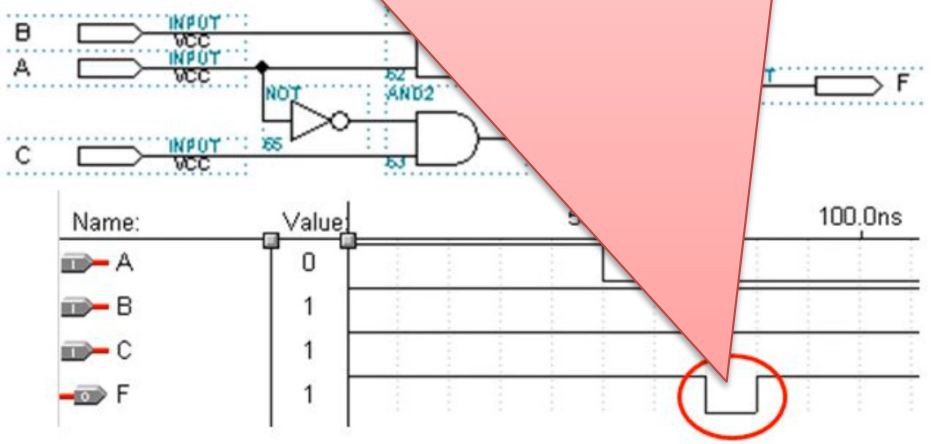
- An input change from ABC = 111 to 011 results in a glitch → output changes from 1 → 0 → 1 again
  - A static hazard: result should have stayed statically at 1

::.. Glitch PUF (1/2)

A purely combinatorial circuit has no internal state: its steady-

Since the exact circuit delays of a particular instance of a combinatorial circuit are influenced by random process variations, the occurrence, the number and the shape of the glitches on its output signals will equivalently be partially random and instance-specific.
When accurately measured, the glitch behavior of such a circuit can be used as a PUF response.

- F = AB + A'C: a = 10ns



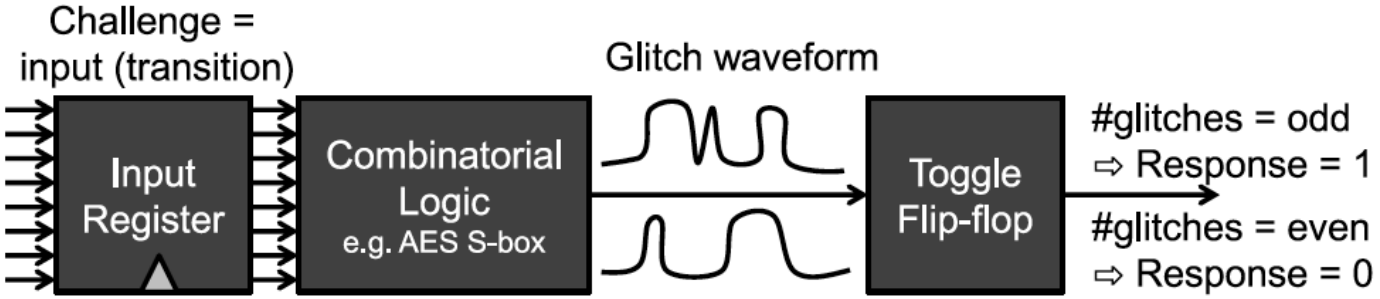
- An input change from ABC = 111 to 011 results in a glitch → output changes from 1 → 0 → 1 again
  - A static hazard: result should have stayed statically at 1

::.. Glitch PUF (2/2)

By simply connecting a combinatorial output to a toggle flip-flop, the value of the toggle flip-flop after the transitional phase will equal the parity of the number of glitches that occurred.

To improve the reliability of the PUF responses, a bit-masking technique is used. During an initial measurement on every instance, unstable response bits are identified as responses that

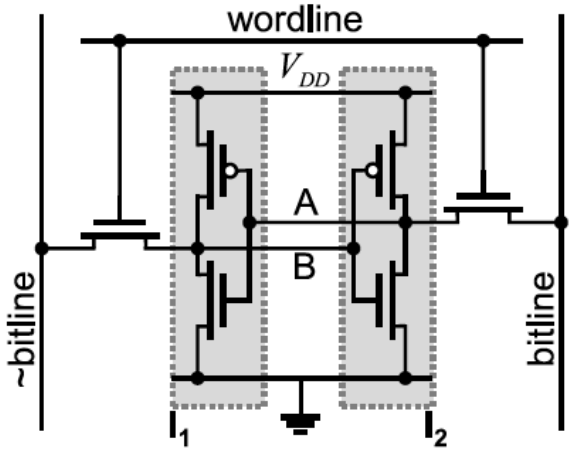
do not produce a stable value on m consecutive evaluations. In later measurements, these response bits are ignored, i.e. they are masked.

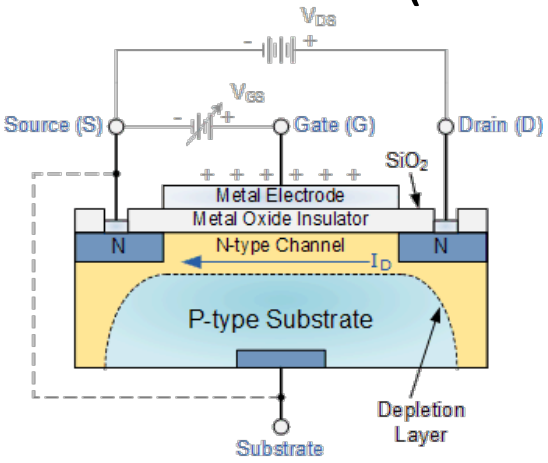


::.. SRAM PUF (1/4)

Static Random-Access Memory (SRAM) is a digital memory technology based on bistable circuits.

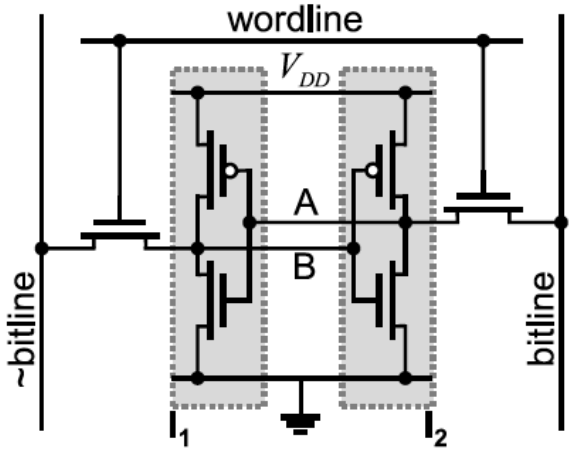
- In a typical CMOS implementation, an individual SRAM cell is built with six transistors (MOSFETs).





::.. SRAM PUF (1/4)

Static Random-Access Memory (SRAM) is a digital memory technology based on bistable circuits.

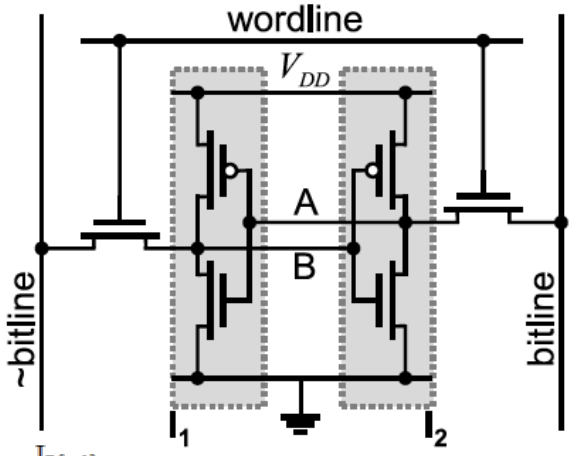


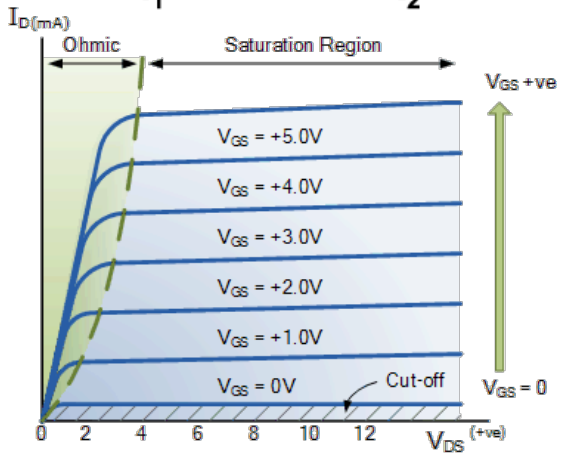
- In a typical CMOS implementation, an individual SRAM cell is built with six transistors (MOSFETs).

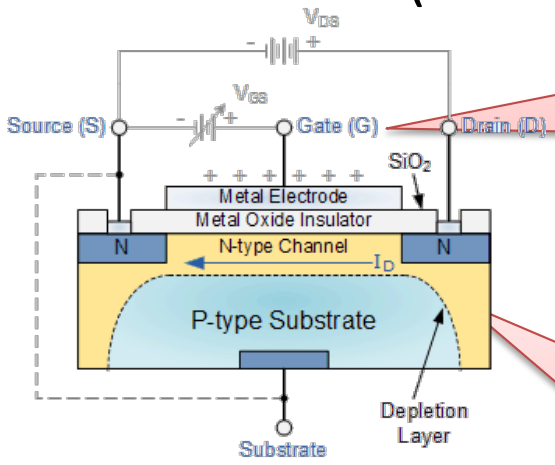
::.. SRAM PUF (1/4)

Static Random-Access Memory (SRAM) is a digital memory technology based on bistable circuits.

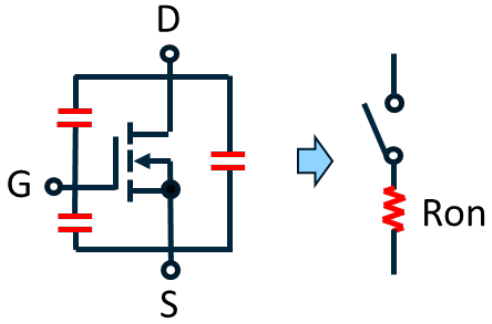
- In a typical CMOS implementation, an individual SRAM cell is built with six transistors (MOSFETs).







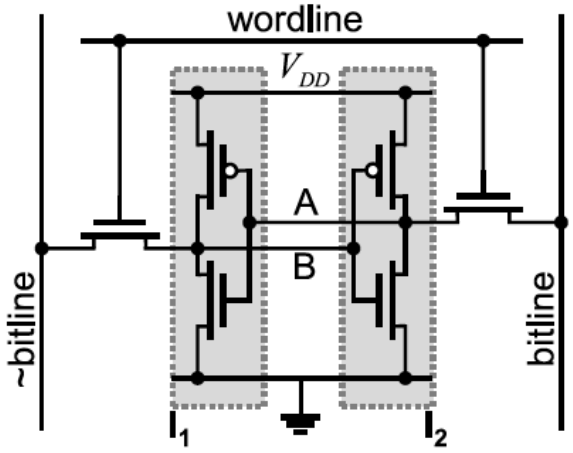
The voltage of the covered gate determines the electrical conductivity of the device.



The element is made of a substrate of semiconductor material, usually silicon, and a metal layer that separates oxygen.

::.. SRAM PUF (1/4)

Static Random-Access Memory (SRAM) is a digital memory technology based on bistable circuits.

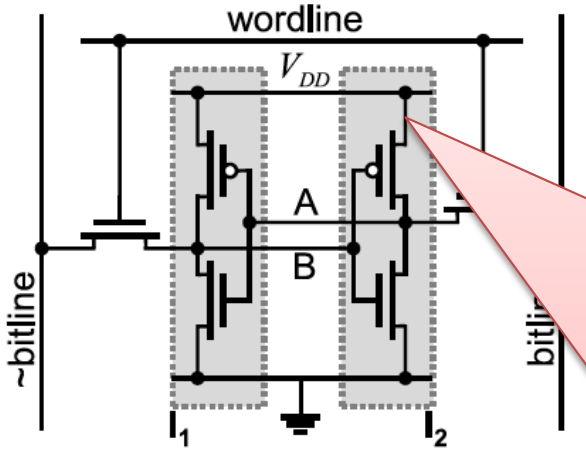


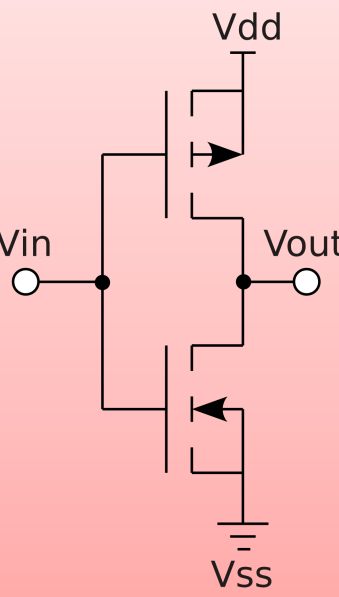
- In a typical CMOS implementation, an individual SRAM cell is built with six transistors (MOSFETs).
- The core is formed by two CMOS inverters, where the output potential of each inverter V_{OUT} is fed as input into the other V_{IN}. This feedback loop stabilizes the inverters to their respective state.

::.. SRAM PUF (1/4)

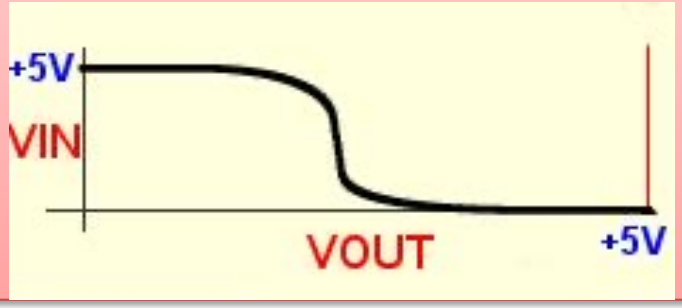
Static Random-Access Memory (SRAM) is a digital memory technology based on bistable circuits.

- In a typical CMOS implementation, an individual SRAM cell is built with six transistors (MOSFETs)



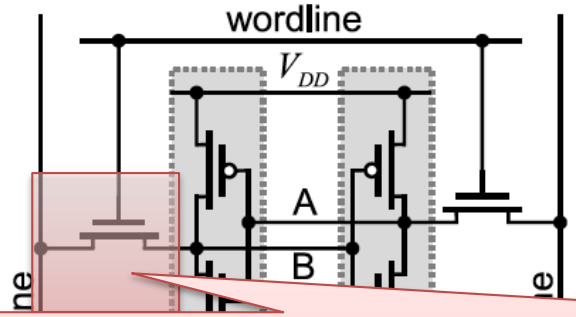


CMOS inverter is composed of two MOSFETs, and it acts as an inverter: when V_{IN} is five volts, V_{OUT} is zero, and vice versa.



::.. SRAM PUF (1/4)

Static Random-Access Memory (SRAM) is a digital memory technology based on bistable circuits.

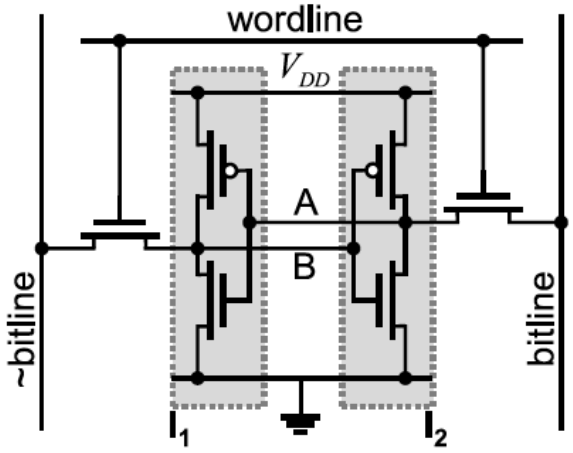


- In a typical CMOS implementation, an individual SRAM cell is built with six transistors (MOSFETs).
- The core is formed by two CMOS inverters, where the output potential of

The access transistors and the word- and bit-lines are used to read and write from or to the cell. In standby mode, the wordline is low, turning the access transistors off. In this state the inverters are in complementary state. To write the data is imposed on the bitline and the inverse data on the inverse bitline. Then the access transistors are turned on by setting the word line to high. As soon as the information is stored in the inverters, the access transistors can be turned off.

::.. SRAM PUF (1/4)

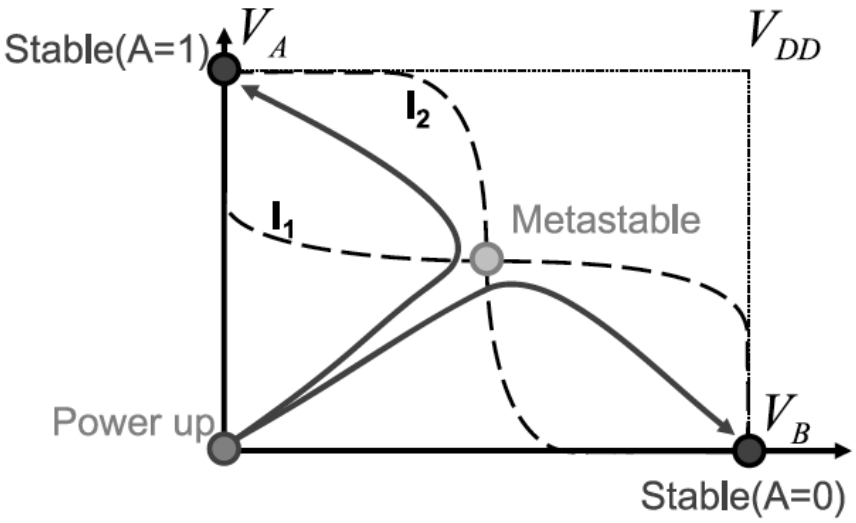
Static Random-Access Memory (SRAM) is a digital memory technology based on bistable circuits.



- In a typical CMOS implementation, an individual SRAM cell is built with six transistors (MOSFETs).
- The core is formed by two CMOS inverters, where the output potential of each inverter V_{OUT} is fed as input into the other V_{IN}. This feedback loop stabilizes the inverters to their respective state.

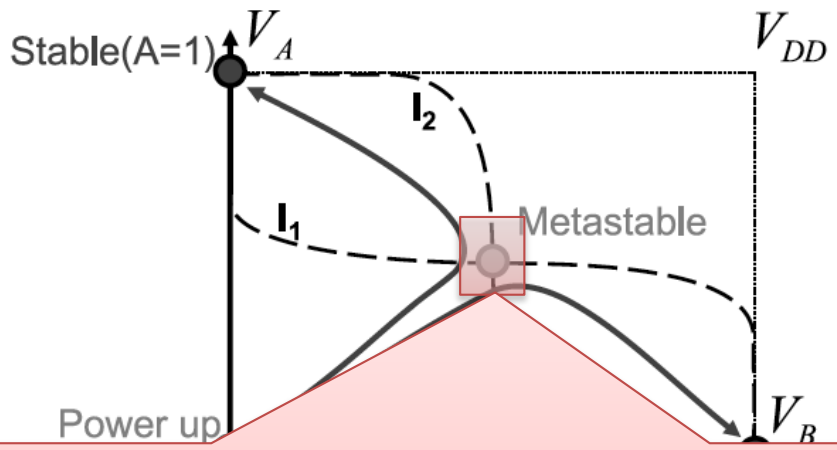
A SRAM can retain its stored information as long as power is supplied. This is in contrast to dynamic RAM (DRAM) where periodic refreshes are necessary or non-volatile memory where no power needs to be supplied for data retention.

::.. SRAM PUF (2/4)



In a logic sense, this circuit has two stable values (bistable), and by residing in one of the two states the cell stores one binary digit. From the voltage transfer curves, it is clear that the cross-coupled CMOS inverter structure has three possible operating points of which only two are stable and one is metastable.

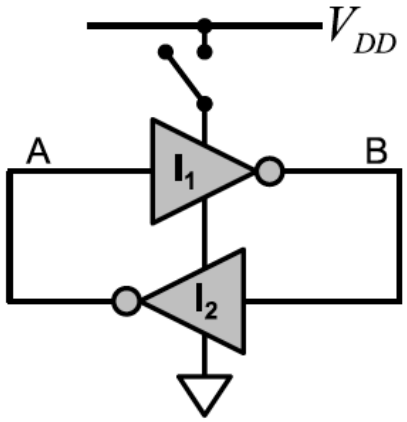
::.. SRAM PUF (2/4)



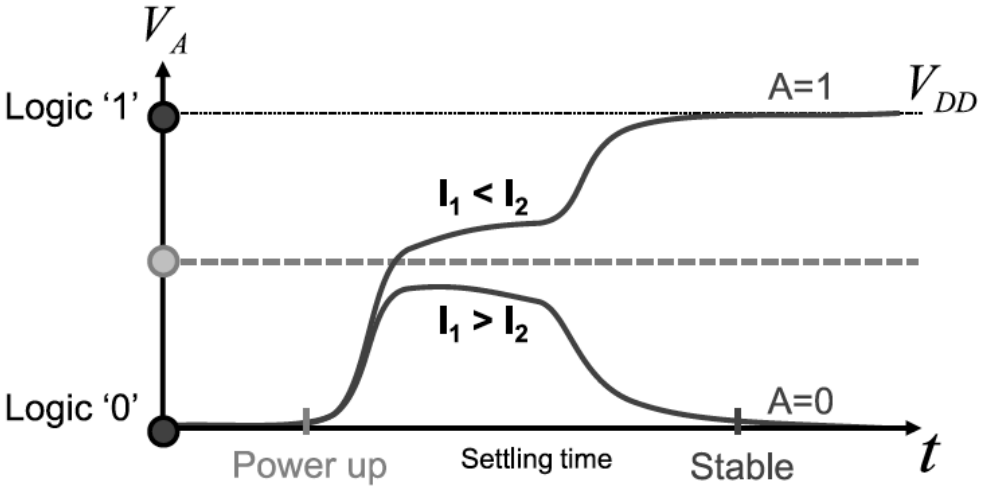
In a logic sense, this circuit has two stable values (bistable), and by residing in one of the two states the cell stores one binary digit. From the voltage transfer curves, it is clear that the cross-coupled CMOS inverter structure has three

Any small deviation from the metastable point is immediately amplified by the positive feedback and the circuit moves away from the metastable point towards one of the two stable points. Since electronic circuits are constantly affected by small deviations due to random noise, an SRAM cell will never stay in its metastable state very long but will quickly end up in one of the two stable states (randomly).

::.. SRAM PUF (3/4)



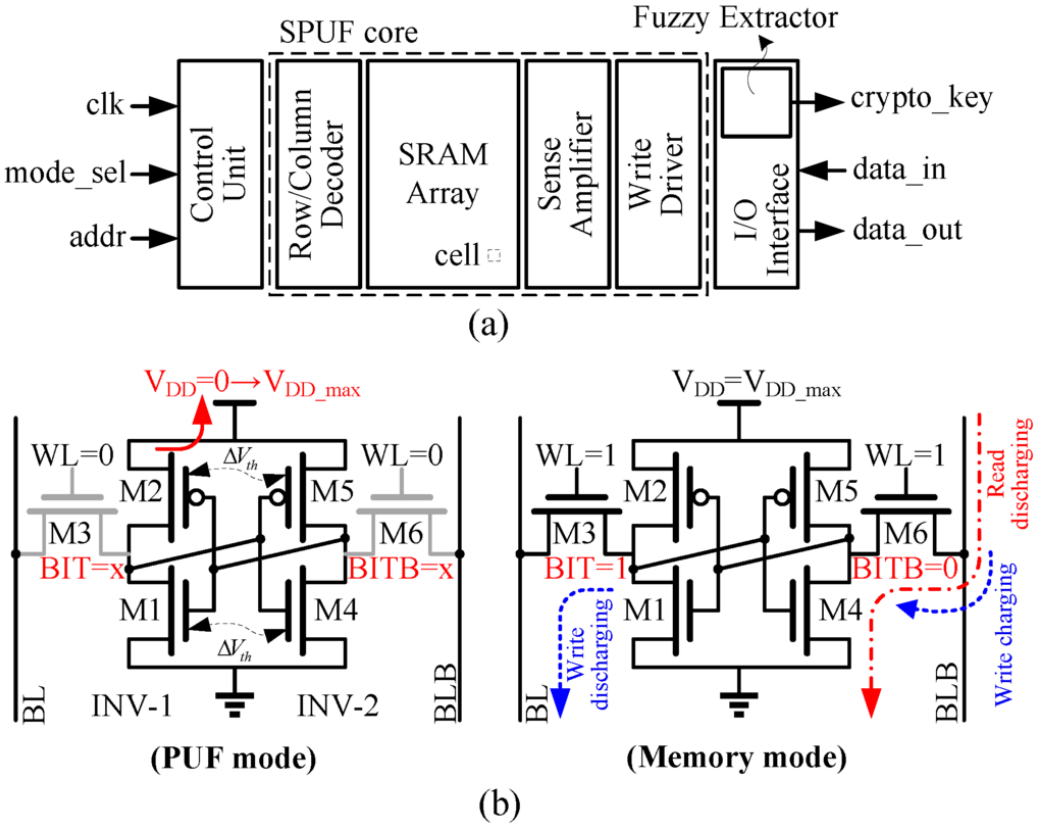
The operation principle of an SRAM PUF is based on the transient behavior of an SRAM cell when it is powered up. The circuit will evolve to one of its operating points, but it is not immediately clear to which one.



The preferred initial operating point of an SRAM cell is determined by the difference in 'strength' of the MOSFETs in the cross-coupled inverter circuit.

The actual difference in strength between the two inverters, or device mismatch, is caused by random process variations in the silicon production process and is hence cell-specific.

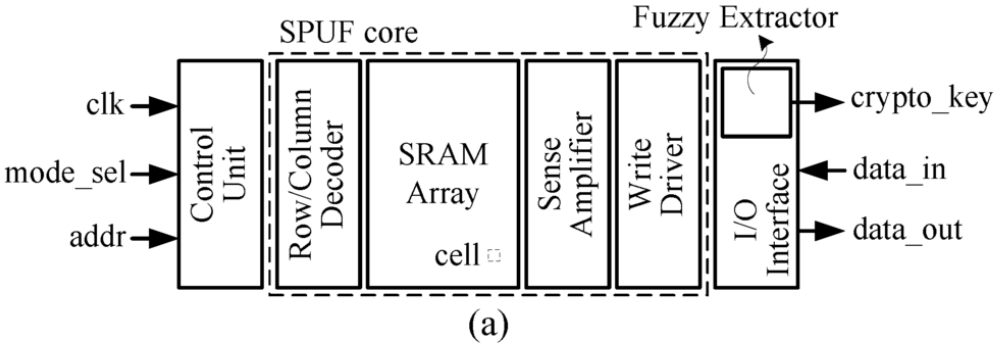
::.. SRAM PUF (4/4)



The core of the SRAM PUF is a two-dimensional array of storage cells, where the horizontal rows are referred to as wordlines and the two vertical lines that run through each cell are called the bitline and complement bitline.

An individual cell can be selected by the row /column decoder with an external input address. One sense amplifier is attached to each bitline pair to accelerate read operation while the write driver circuit improves the access time in write operation.

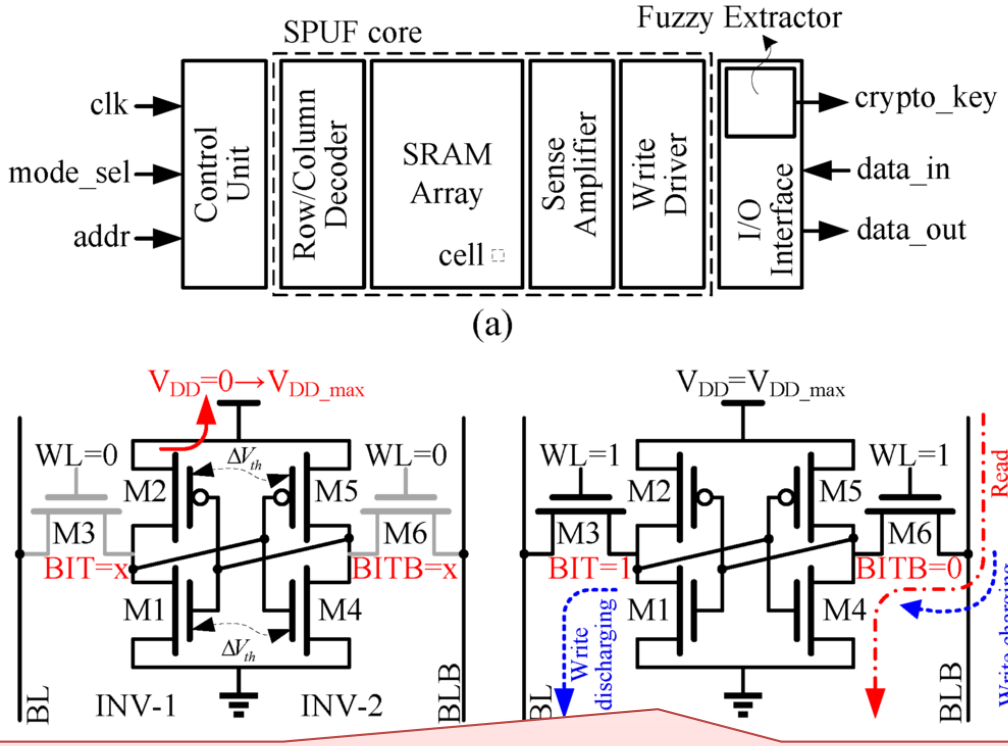
::.. SRAM PUF (4/4)



The core of the SRAM PUF is a two-dimensional array of storage cells, where the horizontal rows are referred to as wordlines and the two vertical lines that run through each cell are called the bitline and

When the SPUF is set to PUF mode, the SRAM PUF is reset and powered up. Due to mismatches and noise between the cross-coupled inverters formed by the transistor pairs, a response bit is generated in each of the SRAM cells. An input challenge is applied to the WL of the addressed cells to turn on the access transistors. The response bits are then read from the bitlines.

::.. SRAM PUF (4/4)

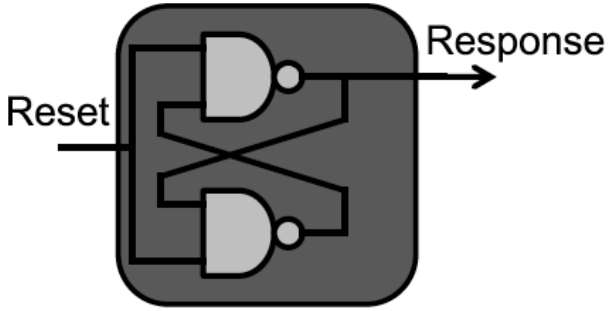


The core of the SRAM PUF is a two-dimensional array of storage cells, where the horizontal rows are referred to as wordlines and the two vertical lines that run through each cell are called the bitline and

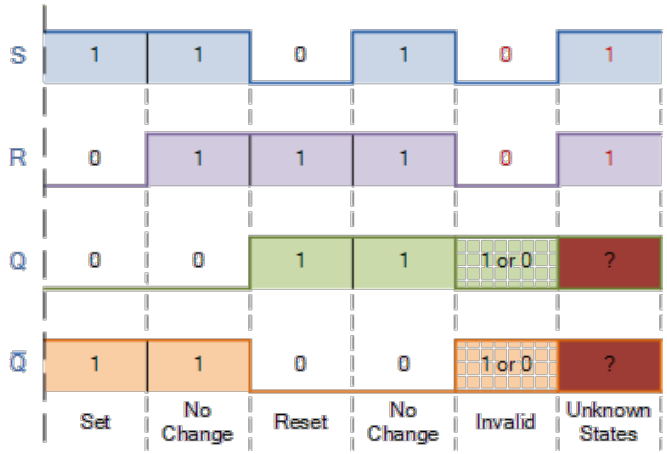
In the memory mode, regular read and write operations are performed on the same SRAM array after the SRAM PUF has been fully powered up. For a read operation, the small voltage difference between the bitlines is amplified by the sense amplifier and the data stored in the addressed cell is transferred to the output. For a write operation, the bit-lines are initialized.

::.. Other Memory-Based PUF (1/5)

Besides SRAM cells, there are a number of alternative, more advanced digital storage elements which are based on the bistability principle: displaying PUF behavior based on random mismatch between nominally matched cross-coupled devices.



- **Latch PUFs**



Two cross-coupled NOR-gates constitute a simple SR latch. By asserting a reset signal, this latch is forced into an unstable state and when released it will converge to a stable state depending on the internal mismatch between the NOR gates.

::.. Other Memory-Based PUF (2/5)

A practical advantage of this latch PUF construction over SRAM PUFs is that the PUF behavior does not rely on a power-up condition, but can be (re)invoked at any time when the device is powered.

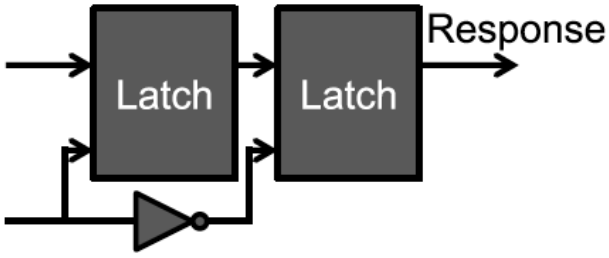
- Secrets generated by the PUF need not be stored permanently over the active time of the device but can be recreated at any time.

- It allows to measure multiple evaluations of each response, which makes it possible to improve reliability through post-processing techniques such as majority voting.

::.. Other Memory-Based PUF (3/5)

- **Flip-Flop PUFs**

Most D flip-flop implementations are composed of a number of latch structures which are used to store a binary state. These internal latch structures cause the PUF behavior of a D flip-flop as for the basic latch PUF at the start-up.

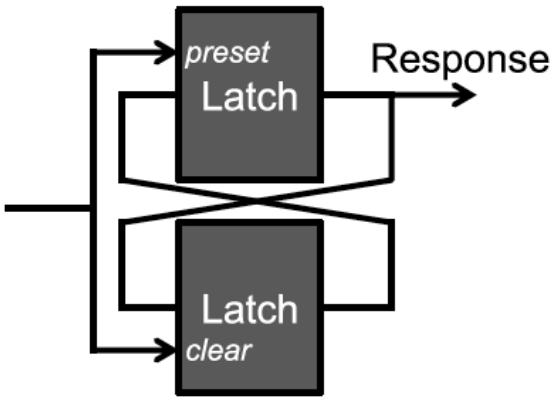


The entropy in the measured flip-flop start-up values is rather limited due to the presence of a strong bias, namely the strong preference of most of the flip-flops to start up with a zero value.

::.. Other Memory-Based PUF (4/5)

- **Butterfly PUFs**

For FPGA platforms, SRAM PUFs are often impossible because the SRAM arrays on most commercial FPGAs (when present) are forcibly cleared immediately after power-up. This results in the loss of any PUF behavior.



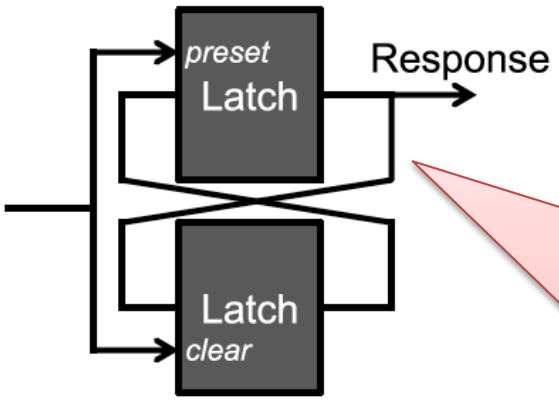
A butterfly consists in mimicking the behavior of an SRAM cell in the FPGA reconfigurable logic by cross-coupling two transparent data latches, forming a bistable circuit.

Using the preset/clear functionality of the latches, this circuit can be forced into an unstable state and will again converge when released.

::.. Other Memory-Based PUF (4/5)

- **Butterfly PUFs**

For FPGA platforms, SRAM PUFs are often impossible because the SRAM arrays on most commercial FPGAs (when present) are forcibly cleared immediately after power-up. This results in the loss of any PUF behavior.



A butterfly consists in mimicking the behavior of an SRAM cell in the FPGA

latches, this circuit ca

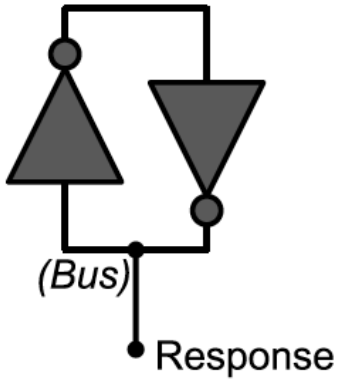
again converge when

Due to the discrete routing options on FPGAs, it is not trivial to implement the cell in such a way that the mismatch by design is small. This is a necessary condition if one wants the random mismatch caused by manufacturing variability to have any effect.

::.. Other Memory-Based PUF (5/5)

- **Buskeeper PUFs**

Another variant of a bistable memory element PUF is based on buskeeper cells, which is a component to a bus line on an embedded system. It is used to maintain the last value which was driven on the line and prevents the bus line from floating.

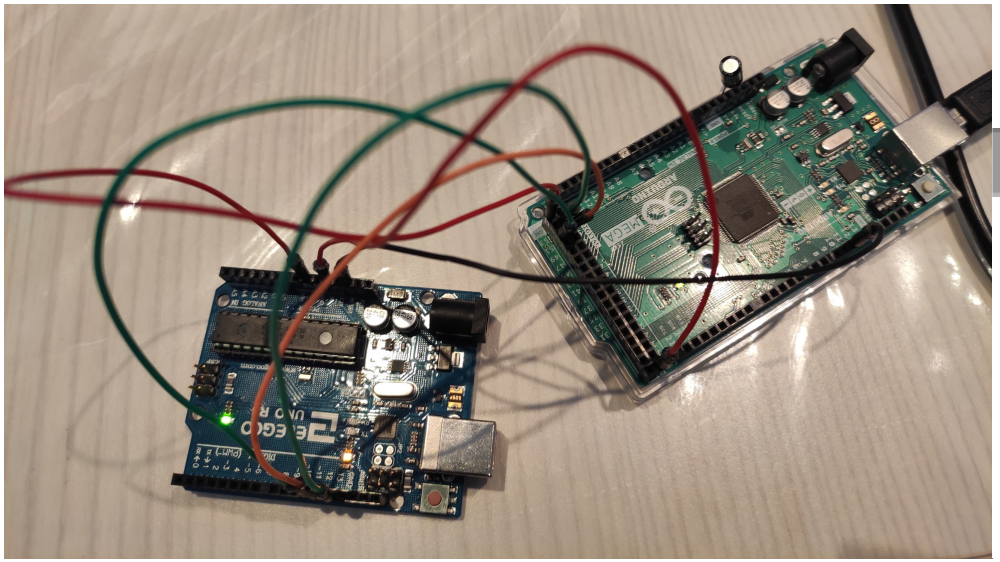


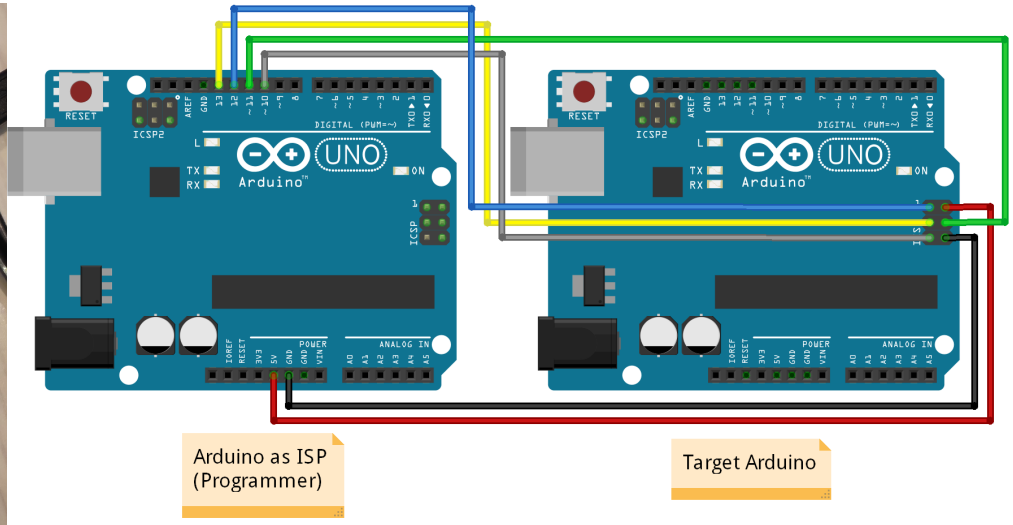
Internally, a buskeeper cell is a simple latch with a weak drive-strength. A cross-coupled inverter structure is again the cause of the PUF behavior of the power-up state of these cells.

An advantage is that a basic buskeeper cell is very small, e.g. in comparison to typical latches and D flip-flops.

::.. SRAM PUF with Arduino (1/2)

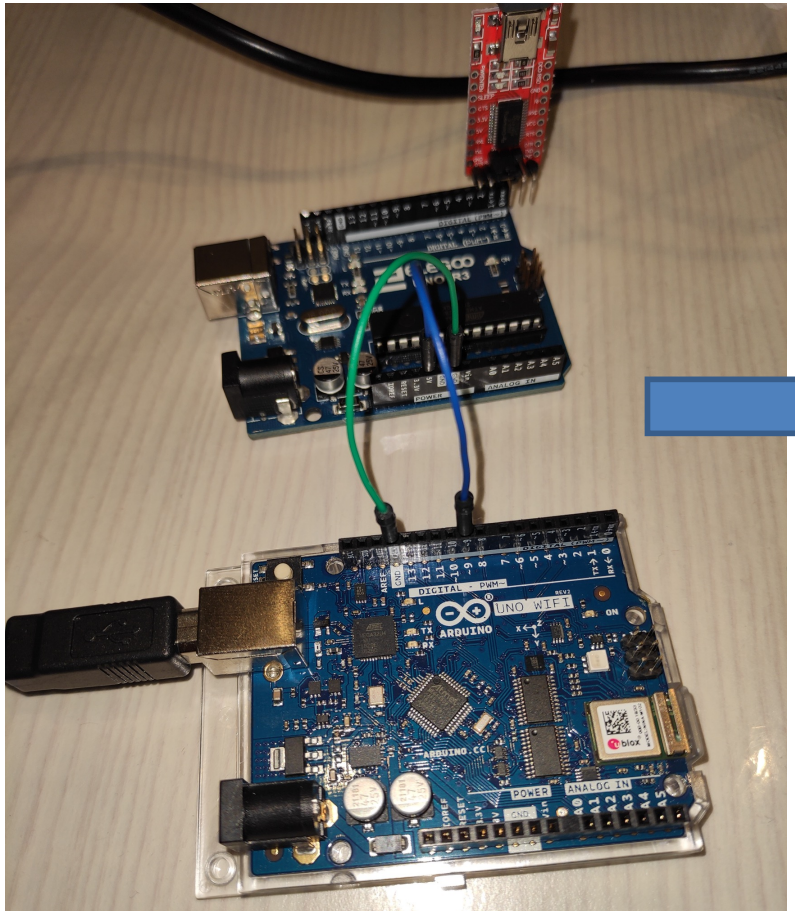
It is possible to read the content of the SRAM in Arduino, which consists of the material needed to build the PUF. The bootloader existing on the Arduino boards initialize the memory content, so it has to be changed. Two Arduino con be connected so that one program the other to load a custom bootloader.





::.. SRAM PUF with Arduino (2/2)

Afterwards, the programmed board can be repeatedly turned on, by using a second board, and the read memory content printed on the serial communication bus.

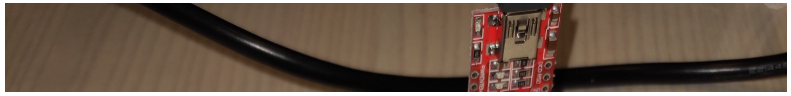


First Read

```
ADB91682B75CFAAA4F738E237B5BBA7D55ACC957C0E0A90969DDE37F62FEA
EF0B351FEB4FD6813CB2356E8BE23CEBF753EF7FD051F755CA943E01BBAC8
9F39F1463FA2F431DF7F69DAFAB73B1969CF3343E86C8D381E3AEDFD09182
78EAB9F4716306FDF080C20246440C9B1D6399D7B7A27D7AD96FC5B413C6F
A2F34395E1234D124E926F67B2AEFB85556EC8DB70EB8D32F240075F4DC63
34BC9ADD58CF83C131ED90975DF92C41626E9D638255FBD8366EDB3DACFE6
0E2CF5AE781AC43EF339A2CFD33EA869BE45EB7E57917A3359BB7EA35ACA7
8B62A2B05A0CF5CFCB3AB0FAB441B1FF77EBA2C5F720F5799F3BF840A6A7A
99058E344118662399CB2BDA00FAA8AE7568CA3950D6B643611980EEB09F3
9B266F85D7221AF9F8DE55A73FBDC5FFBB0F3DBF6D7C5E6C3B1FBF708FAA5
204377E8967F411EAEAABAB12898B153EFFAEEBDBE769E6527FE7E558AFF2
5547AD5B5457A1D323E5F9AE685C3FEE66C5139F0BE0E21B469B17A67209E
7828BAED90781442E22D1E0184A59BF57313D04C5A21DC568259E361DA6BE
6FE44DE25F15B6117346B22DECDF905970F3EFF03462D30A29E517870737E
22ADB45F3D30DEEB7D3E2FB2979D4CDFCB1E106BC1EDF28AAB935C90D41AC
B94CEC5BE46E0D52A139BA4712B575BD905BBA5DE8EFE22D7ECF023D27E87
589C569FF68D23BD78124C56A612ED6B3479DCF83D57DE59FBAFCBFB37FC7
356C41D2FEF8862929D8C3E5D8B823EB1E8F0145FFC406C1FAB4A0F90898D
238AF0968BF4B6719ED37961D0253EC04F736771395ABBD21FA7867FC29E4
774F8E0A477F0F7060DE67EB31517012E105E466E135719EA566FF73FD292
7F9DA7B5CAEB1935D338DE76FFEB6294148CB73B13DACA635239AC1CCA06B
413DC1E80FF882A04B896C4BFDA9DD33EBA8778FEB850D61CBA953DFD4C3F
01DE827ED91D0B7E2588C3DA9E30CEA85D9FE16FC8D06237D3269D1FE22CC
ED378A0A62F63BD342EF59E5BD587B306B8A8F3A4897C46DAAADF89135BAB
A52FAFF177D4FBD2BCC3F42AA14A59D2B73B3EC7458367ABBE43CEFB6C7F5
E6EDE96C5265FC945F287016C472D70BF29DA4EF4F6654436F8382A6607E5
F1FF232C70F68006F82AF7BCBDEFA7A8B8DECFB4C5341D79E52AB7F61A046
```

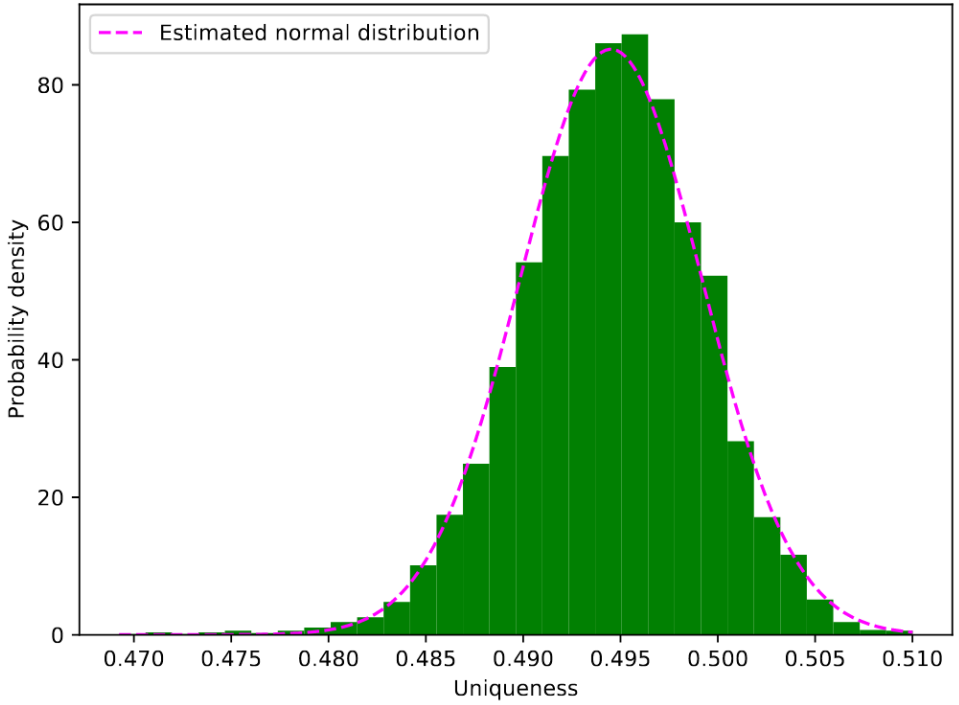
::.. SRAM PUF with Arduino (2/2)

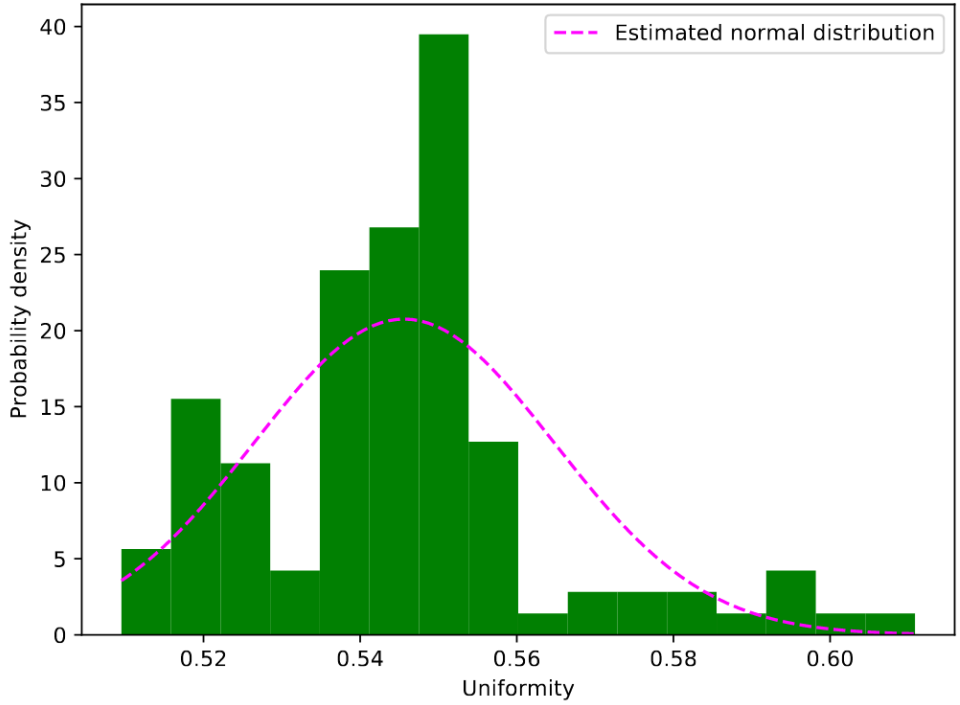
Afterwards, the programmed board can be repeatedly turned on, by using a second board, and the read memory content printed on the serial communication bus.



First Read

```
ADB91682B75CFAAA4F738E237B5BBA7D55ACC957C0E0A90969DDE37F62FEA
EF0B351FEB4FD6813CB2356E8BE23CEBF753EF7FD051F755CA943E01BBAC8
9F39F1463FA2F431DF7F69DAFAB73B1969CF3343E86C8D381E3AEDFD09182
```

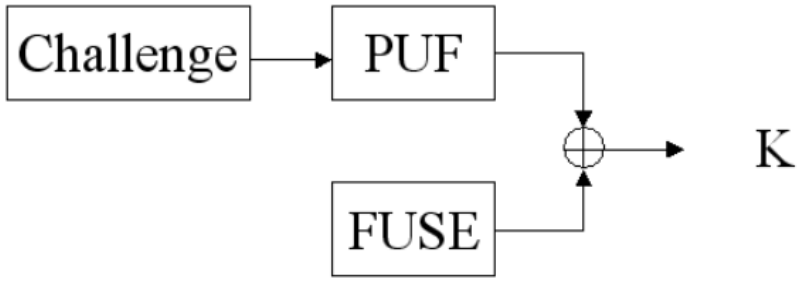




```
E6EDE96C5265FC945F287016C472D70BF29DA4EF4F6654436F8382A6607E5
F1FF232C70F68006F82AF7BCBDEFA7A8B8DECFB4C5341D79E52AB7F61A046
```

::.. Physically Obfuscated Keys

The only condition for a POK is that a key is permanently stored in a ‘physical’ way instead of a digital way, which makes it hard for an adversary to learn the key by a probing attack. Additionally, an invasive attack on the device storing the key should destroy the key and make further use impossible, hence providing tamper evidence.



To be able to set a POK to a chosen value, the output of the PUF – which cannot be chosen – can be combined to some fuse (or any data stored in the EEPROM) via an exclusive-or operation.

::.. Controlled PUFs (1/2)

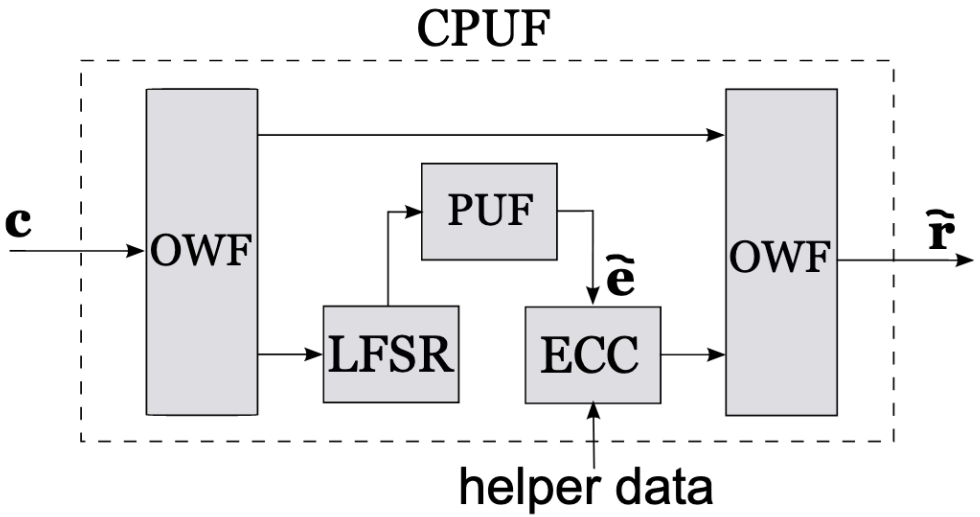
A controlled PUF (CPUF) is a mode of operation for a PUF in combination with other (cryptographic) primitives. A PUF is said to be controlled if it can only be accessed via an algorithm which is physically bound to the PUF in an inseparable way.

- A (cryptographic) hash function to generate the challenges of the PUF can prevent chosen-challenge attacks, e.g. to make model-building attacks more difficult.
- An error-correction algorithm acting on the PUF measurements makes the final responses much more reliable, reducing the probability of a bit error in the response to virtually zero.
- A (cryptographic) hash function applied on the error-corrected outputs effectively breaks the link between the responses and the physical details of the PUF measurement.

::.. Controlled PUFs (2/2)

- The hash function generating the PUF challenges can take additional inputs, e.g. allowing to give a PUF multiple personalities. This might be desirable when the PUF is used in privacy-sensitive applications.

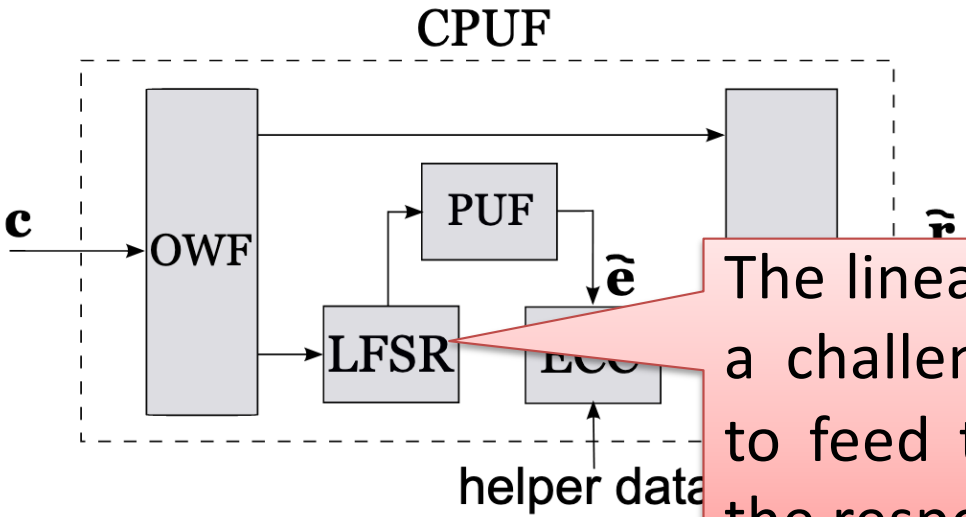
In a generic model, the challenge is pre-processed while the response is then post-processed, all via one-way function (OWF) primitives. Before the OWF, the unreliable responses are marked to be corrected. However, the ECC logic and storing of helper data are always expensive and opens up vulnerabilities.



::.. Controlled PUFs (2/2)

- The hash function generating the PUF challenges can take additional inputs, e.g. allowing to give a PUF multiple personalities. This might be desirable when the PUF is used in privacy-sensitive applications.

In a generic model, the challenge is pre-processed while the response is then post-processed, all via one-way function (OWF) primitives. Before the OWF, the unreliable



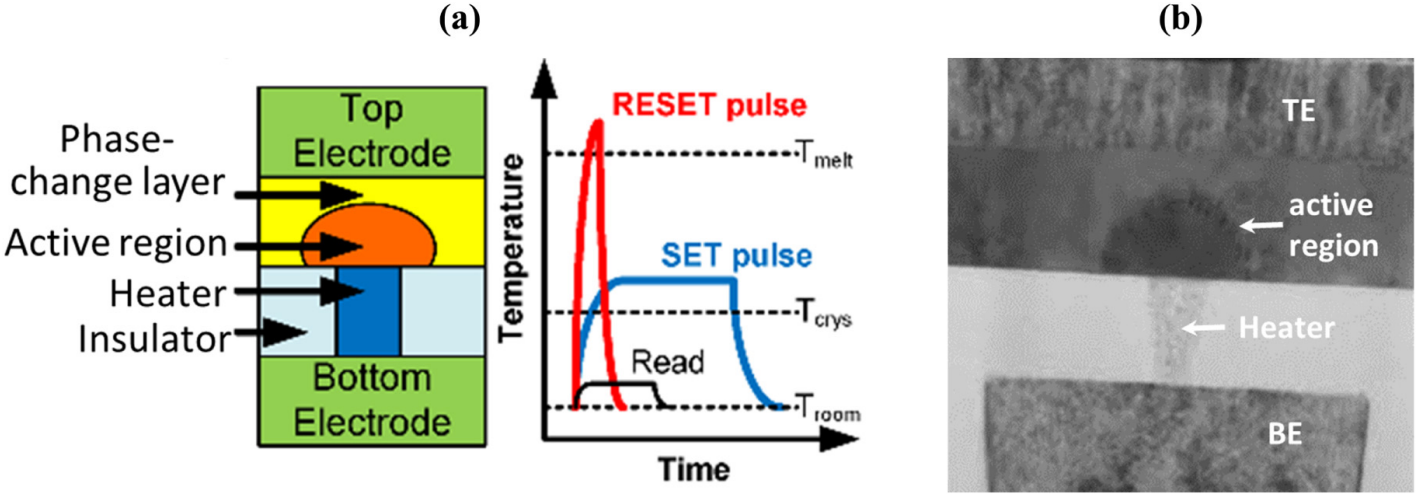
The linear feedback shift register (LFSR) takes a challenge and generates n sub-challenges to feed the PUF and have a but composing the response.

up vulnerabilities.

::.. Reconfigurable PUFs (1/3)

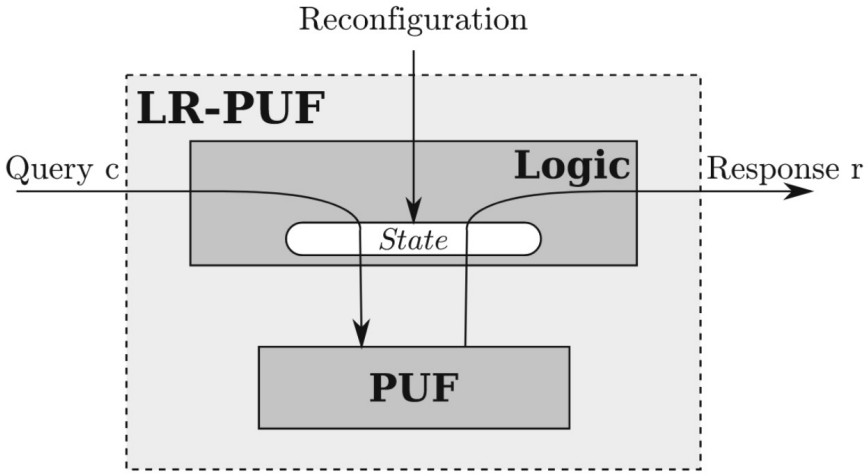
Reconfigurable PUFs (RPUFs) consists in extending the regular challenge-response behavior of a PUF with an additional operation called reconfiguration, to partially or completely change challenge-response behavior of the PUF in a random and preferably irreversible manner, leading to a new PUF.

- A NVRAM called phase change memory is made of the so-called 'mushroom' (or 'lance') cell allowing to be amorphized and re-cystallized, thus also limiting the switching current and power.



::.. Reconfigurable PUFs (2/3)

Writing to such a memory amounts to physically altering the phase of a small cell from crystalline to amorphous or somewhere in between, and it is read out by measuring the resistance of the cell. The exact measured resistances can be used as responses, and rewriting the cells will change them in a random way.



- Another construction is logically reconfigurable PUF (LRPUF). By having a portion of non-volatile memory storing a state and state-dependent input- and output-transformations applied to the response of a classic PUF.

::.. Reconfigurable PUFs (3/3)

Drawbacks of logical reconfigurable PUF than physical ones:

- The reconfiguration of an LRPUF is not truly irreversible, since if an old internal state is restored, the same challenge-response behavior will reappear;
- the security of the logical reconfiguration relies on the integrity of the state memory, which needs to be guaranteed by independent physical protection measures.

However, their relatively easy construction in comparison to the rather exotic physically reconfigurable PUF proposals makes them immediately deployable.

::.. PUF-Based Identification (1/11)

An identity can be

- Inherent, i.e., based on an entity-specific characteristic that arises in the creation process of the entity;
- Assigned, e.g., a barcode or a password.

The inherency of its instance-specific behavior was indicated as one of the key conditions for a construction to be called a PUF.

Unique bit strings programmed in a non-volatile memory embedded on the chip were until recently the only way of identifying a specific chip in a digital interaction.

With silicon PUF technology, it is now possible to also use inherent unique features of a silicon chip for instance identification.

::.. PUF-Based Identification (2/11)

Identification techniques based on assigned as well as on inherent identities typically work in two phases.

The first phase is different for both types:

- For assigned identities, the first (provisioning) phase consists of providing every entity with a permanent unique identity.
- For inherent identities, the first (enrollment) phase consists of collecting the inherent identities of every entity that needs to be identified.

The second (identification) phase is very similar for both types and consists of an entity presenting its identity, either assigned or inherent, when requested.

::.. PUF-Based Identification (3/11)

The differences between provisioning and enrollment highlight interesting practical advantages:

- A unique assigned identity needs to be generated before it is assigned to an entity. To ensure identity uniqueness, the provisioning party either needs to keep state, e.g. using a monotonic counter, or requires a randomness source, e.g. a true random-number generator. For inherent identities this is not required, since their uniqueness results from the creation process of the entities.
- When assigning an identity, the provisioning party needs to make permanent (or at least non-volatile) physical changes to each entity. This needs to be supported by the entity's construction, with increasing costs. Evidently, inherent identities do not require additional storage capabilities.

::.. PUF-Based Identification (4/11)

- Enrollment (reading an identity) is generally less intrusive than provisioning (writing an identity); hence it can be done faster and with a higher reliability. On the downside, there is no direct control over the actual values taken by inherent identifiers. This is an issue if one wants to assign a meaning to an identifier value.

A particular trait of inherent identifying features is that they show fuzzy random behavior. We say that a random variable, like a PUF response or a biometric feature, shows fuzzy behavior if:

1. It is not entirely uniformly distributed, and
2. it is not perfectly reproducible when measured multiple times.

::.. PUF-Based Identification (5/11)

For PUF responses,

- fuzzy characteristics are caused by the physical nature of their generation. Random physical processes that introduce entity-specific features during manufacturing are typically not uniformly distributed.
- The response evaluation mechanisms of a PUF construction are subject to physical noise and environmental conditions which cause a non-perfect reproducibility of a PUF response value.

Assigned identities on the other hand are typically not fuzzy: the provisioning party can make sure that the assigned identities are generated from a uniform distribution, and an entity can typically reproduce its assigned identity with near-perfect reproducibility.

::.. PUF-Based Identification (6/11)

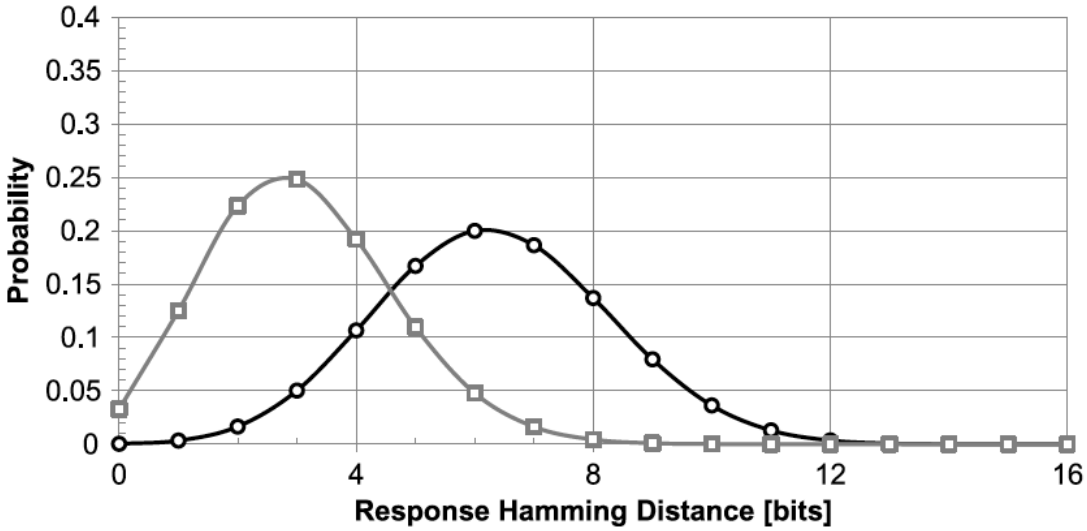
The fuzziness of a PUF response is most clearly depicted by its inter- and intra-distance distributions. When we consider binary response vectors and fractional Hamming distance as a distance metric, then perfectly uniformly random responses would have an expected inter-distance of exactly 50%.

- None of the existing intrinsic PUF constructions meet this condition, although some have an average inter-distance very close to 50%.
- No intrinsic PUF exhibits perfect reproducibility with a fixed intra-distance of 0 %, and many are only reproducible up to an average intra-distance of 10 % or even more.

PUF property of identifiability: a PUF class exhibits identifiability if with high probability its responses' intra-distances are smaller than their inter-distances.

::.. PUF-Based Identification (6/11)

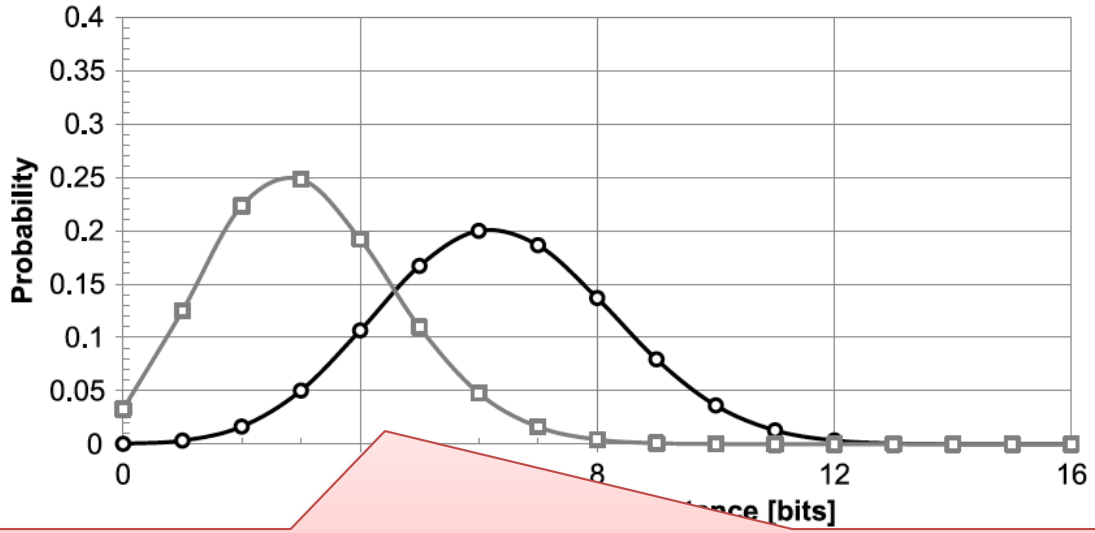
Let us compute the identifying power of a PUF's responses based on their intra- and inter-distance distributions when considering a D flip-flop PUF which produces a 16-bit response.



To estimate both distance distributions, we assume the binomial probability estimators resulting from the experimental analysis of the PUF.

::.. PUF-Based Identification (6/11)

Let us compute the identifying power of a PUF's responses based on their intra- and inter-distance distributions when considering a D flip-flop PUF which produces a 16-bit response.

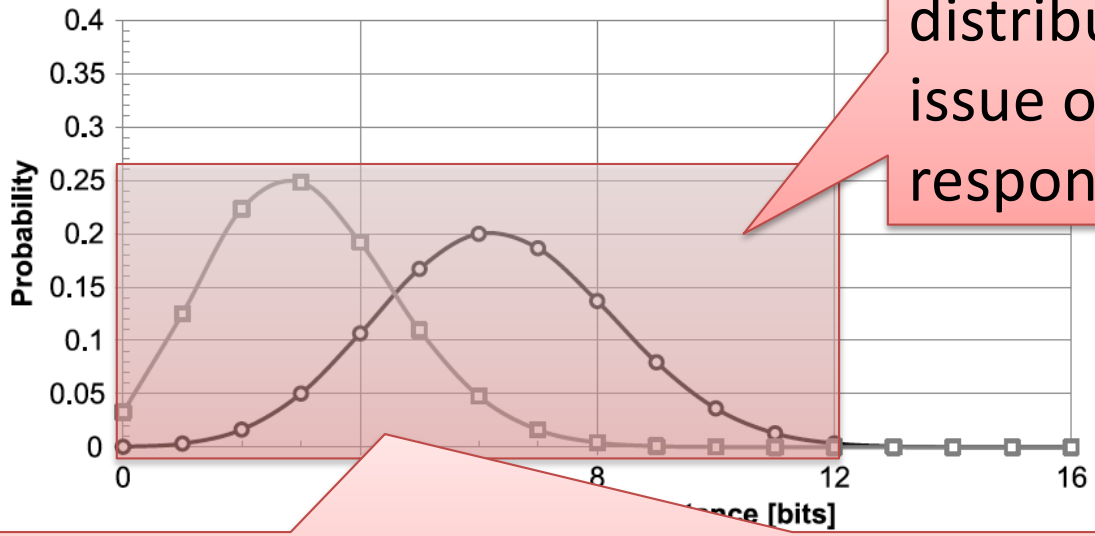


To estimate both distance distributions, we assume the binomial probability estimators resulting from the experimental analysis of the PUF.

This PUF construction exhibits some level of identifiability, since the expected intra-distance is noticeably smaller than the expected inter-distance.

::.. PUF-Based Identification (6/11)

Let us compute the identifying power of a PUF's responses based on their intra- and inter-distance distributions when considering a D flip-flop PUF which produ



There is also a significant overlap between the curves of the two distributions, which points out the issue of identification based on fuzzy responses.

estimators resulting from the experimental analysis of the PUF.

This PUF construction exhibits some level of identifiability, since the expected intra-distance is noticeably smaller than the expected inter-distance.

::.. PUF-Based Identification (7/11)

When an observed distance between responses from the enrollment and the identification phase falls in this overlapping region, it can be a result of intra-distance, in which case it is the same entity, or of inter-distance, in which case it concerns a different entity, and there is no way of distinguishing between the two cases.

- Identification threshold: a distance threshold is set:
  - Distances smaller or equal to this threshold are assumed to be intra-distances between responses from a single entity;
  - Distances above this threshold are assumed to be inter-distances between responses from different entities.

A fuzzy identification system based on such a pragmatic identification threshold is not 100 % reliable.

::.. PUF-Based Identification (8/11)

When comparing a presented entity response to a response from the enrollment list, four possible situations can arise:

1. The presented entity is the same entity that produced the enrolled response, and manages to reproduce the enrolled response with an intra-distance smaller than the identification threshold. True acceptance: The presented entity is correctly identified.

2. The presented entity is the same entity that produced the enrolled response, but is not able to reproduce the enrolled response with an intra-distance smaller than the identification threshold. False rejection: The presented entity is mistakenly rejected.

::.. PUF-Based Identification (9/11)

3. The presented entity is not the same entity that produced the enrolled response, but happens (by chance) to produce a response whose inter-distance to the enrolled response is smaller than the identification threshold. False acceptance: The presented entity is mistakenly identified.
4. The presented entity is not the same entity that produced the enrolled response, and it produces a response whose inter-distance is larger than the identification threshold. True rejection: The presented entity is correctly rejected.

False rejections and false acceptances are both undesirable for a practical identification system. The probability that a random identification attempt results in one of these cases is respectively expressed as the false rejection rate or FRR and as the false acceptance rate or FAR of the system

::.. PUF-Based Identification (10/11)

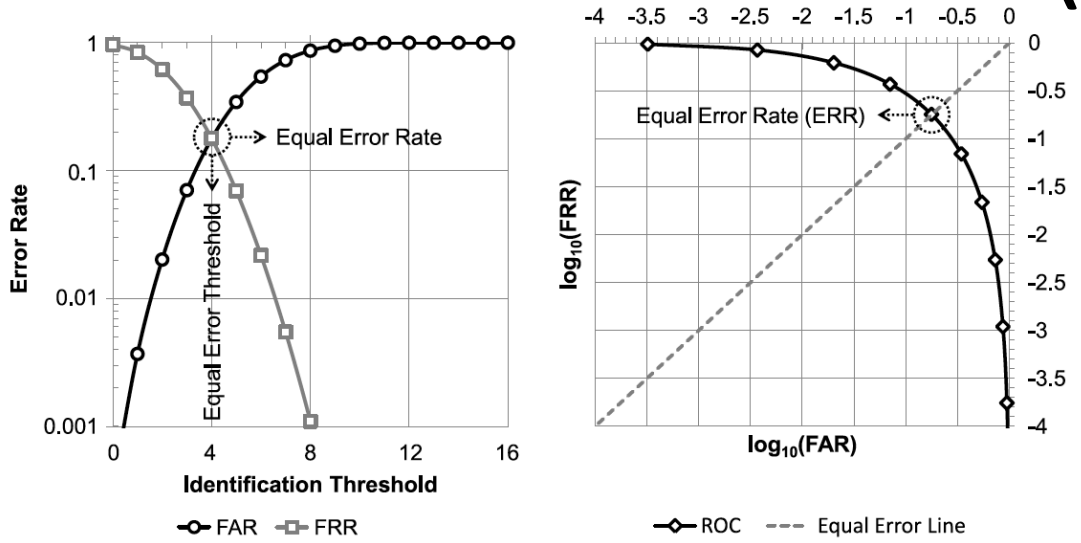
FAR expresses the security of an identification system: a low FAR means that there is little risk of misidentification which could lead to security issues.

FRR on the other hand expresses the robustness or usability of a system, as it expresses the risk of wrongfully rejecting legitimate entities, which would be very impractical.

Both FAR and FRR need to be as small as possible, but it is evident that they cannot be both minimized at the same time. An acceptable trade-off between security and usability needs to be made depending on the choice for the identification threshold value.

- A high threshold minimizes the risk of a false rejection but increases the likelihood of false acceptances, and vice versa for a low threshold.

::.. PUF-Based Identification (11/11)



(a) FAR and FRR plots (b) ROC curve plot

The FAR and FRR plots can be used to find a suitable trade-off meeting the application requirements.

In some applications more care is given to security than to usability, or vice versa. A more convenient way for assessing the FRR-vs-FAR trade-off is the plot of FRR as a function of FAR by using the receiver-operating characteristic or ROC plot of the system. In a ROC plot, the trade-off is found as the intersection with the identity function.

::.. PUF-Based Authentication (1/7)

When an entity wants to authenticate itself to an another party, typically called the verifier, it needs to provide, besides plain identification, also corroborative evidence of its presented identity, i.e. evidence which could only have been created by that particular entity.

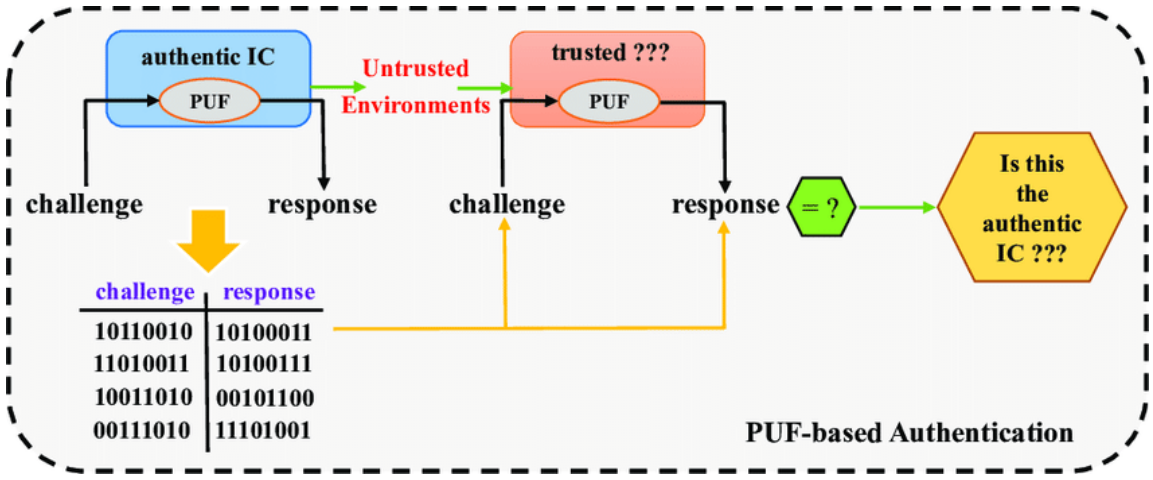
An entity typically achieves this goal by proving to the verifier that it knows, possesses or contains a particular secret which only that entity can know, have or contain. In addition, the entity also needs to convince the verifier that it was actively, i.e. at the time of authenticating, involved in creating that evidence.

How can the PUF be used in device authentication?

::.. PUF-Based Authentication (2/7)

- **PUF-Based Challenge-Response Authentication**

The basic PUF-based challenge-response entity authentication scheme consists of two phases, enrollment and verification:

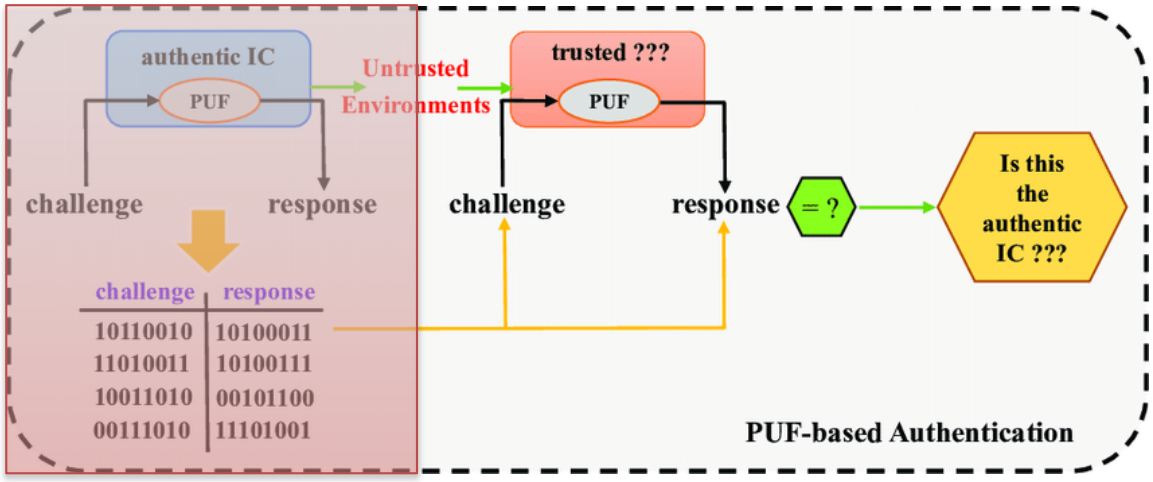


::.. PUF-Based Authentication (2/7)

- **PUF-Based Challenge-Response Authentication**

The basic PUF-based challenge-response entity authentication scheme consists of two phases, enrollment and verification:

1. Before deployment, every entity goes through enrollment by the verifier. The verifier records the identity ID, and collects a significant subset of challenge-response pairs stored in the verifier's database, indexed by the entity's ID.

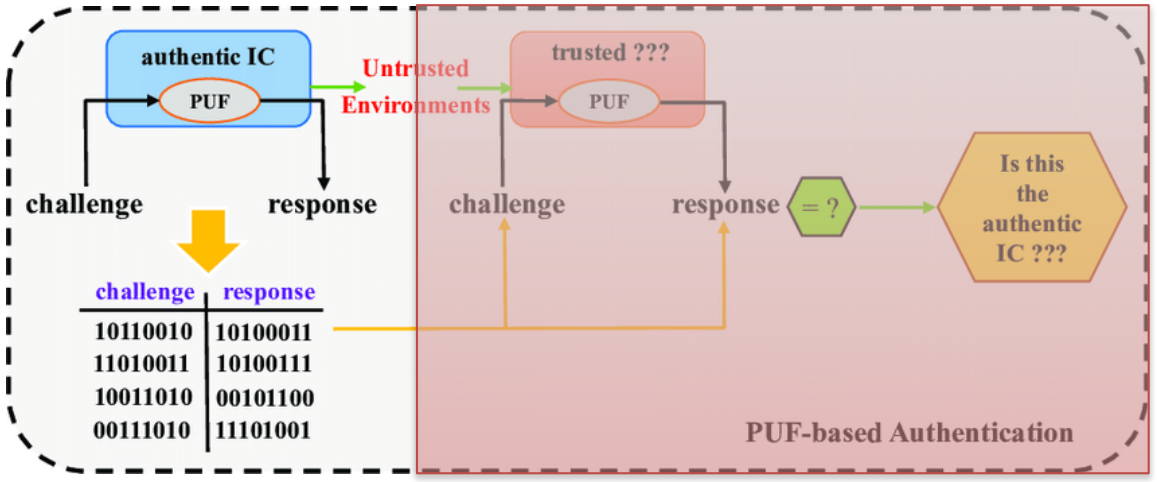


::.. PUF-Based Authentication (2/7)

- **PUF-Based Challenge-Response Authentication**

The basic PUF-based challenge-response entity authentication scheme consists of two phases, enrollment and verification:

2. During the verification phase, an entity sends its ID. The verifier looks up the ID in its DB, and selects a challenge-response. The PUF challenge is sent to the entity, which replies with the obtained response checked by the verifier.



::.. PUF-Based Authentication (3/7)

The correctness of this authentication scheme is ensured by the fact that PUF responses are reproducible over time, up to a small intra-distance. For the security of the authentication, the verifier relies on the fact that PUFs are unique and unclonable, and only the genuine PUF can with high probability reproduce a close response to a previously unobserved and random challenge.

This basic protocol exhibits a number of major shortcomings and drawbacks:

- CRPs cannot be reused in order to avoid replay attacks, and a used pair is therefore removed from DB after the protocol finishes. The verifier needs to store a large number of pairs for each entity, and maintaining such a large database is a significant effort. Not all PUF has a big challenge-response set

::.. PUF-Based Authentication (4/7)

- When the stored challenge-response pairs in DB of an entity run out, the entity can no longer be authenticated by the verifier.
- The basic protocol only provides authentication of an entity to the verifier; no mutual authentication can be supported without significantly extending the basic protocol.
- The basic protocol is only secure for truly unclonable PUFs, but PUFs which can be cloned in a mathematical sense do not offer secure authentication.

Unfortunately, no intrinsic PUF candidates currently exist which meet the very strong requirement of mathematical unclonability needed to make the basic challenge-response authentication protocol secure.

::.. PUF-Based Authentication (5/7)

- **PUF-Based Mutual Authentication**

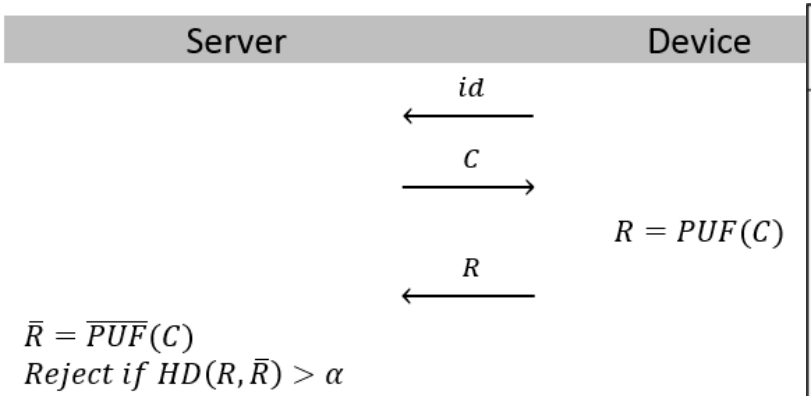
The amount of unpredictability in the challenge-response behavior of an intrinsic PUF instance is strictly limited, and increasing it comes at a high relative implementation cost.

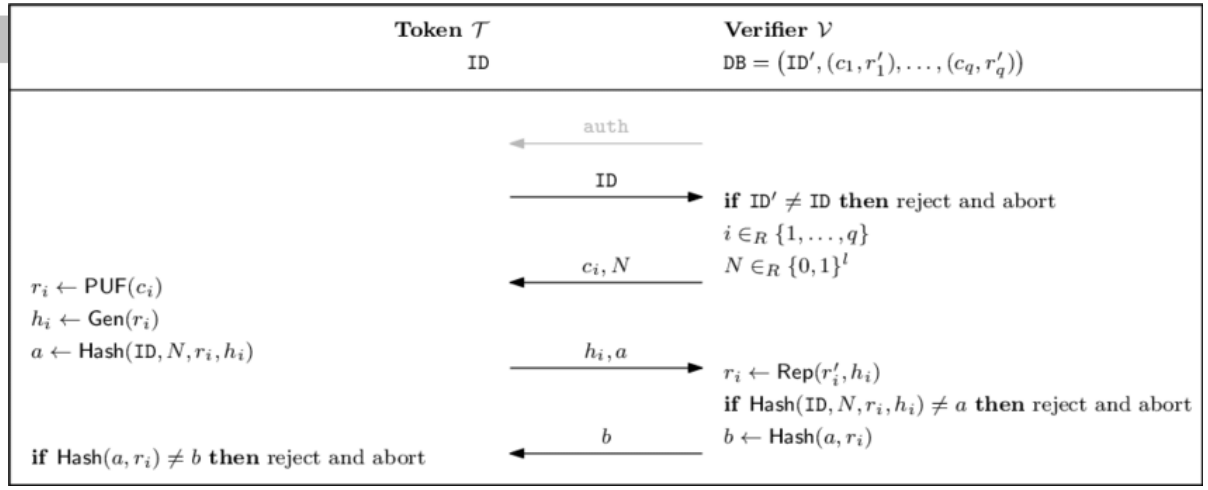
It is not recommendable to publicly disclose challenge-response pairs in the protocol's communications, as the basic protocol does, since every disclosed pair significantly reduces the remaining unpredictability of the PUF's behavior.

In classic cryptographic challenge-response authentication protocols based on symmetric-key techniques, an entity does not authenticate itself by disclosing its secret key, but it only proves its knowledge of the secret key.

::.. PUF-Based Authentication (6/7)

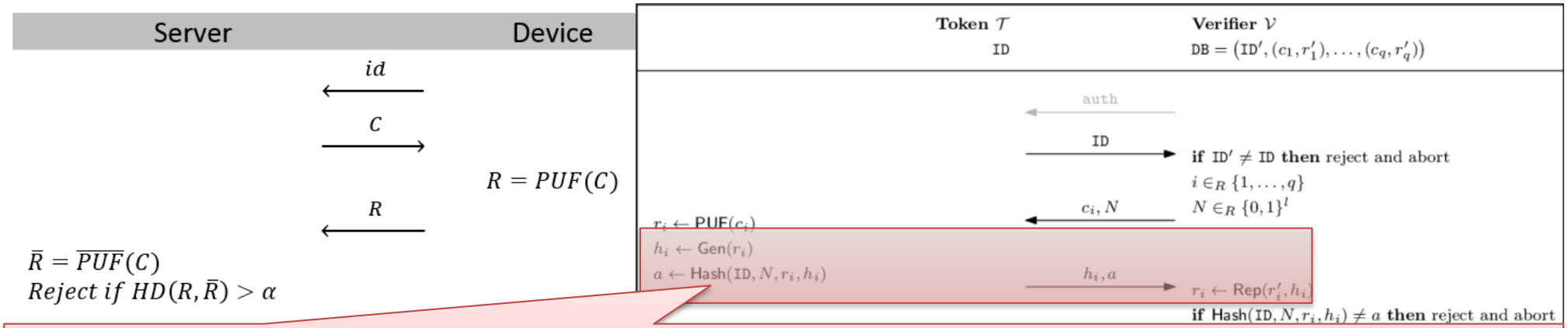
In PUF-based authentication, an entity demonstrates its possession of a PUF instance by showing that it can calculate a function evaluation which takes an unpredictable PUF response as input, without fully disclosing the unpredictable nature of the response in the result of this evaluation.





::.. PUF-Based Authentication (6/7)

In PUF-based authentication, an entity demonstrates its possession of a PUF instance by showing that it can calculate a function evaluation which takes an unpredictable PUF response as input, without fully disclosing the unpredictable nature of the response in the result of this evaluation.



Since PUFs are inherently noisy, they are typically combined with fuzzy extractors, i.e., a secure sketch mapping similar PUF responses to the same value. A hash function is used to generate the proof, as it does not disclose any significant information about the response value.

::.. PUF-Based Authentication (7/7)

Due to the physical unclonability of the deployed PUFs, an impersonation attempt of an entity carrying a different PUF will fail with high probability. The actual false acceptance rate of the overall protocol also depends on the collision resistance of the used hash function and could be considerably higher.

The random nonces respectively generated by the entity and the verifier preclude replay attacks from both sides, by introducing freshness in the protocol communications. An adversary trying to replay protocol messages in order to impersonate the verifier will fail, as well as an adversary trying to impersonate an entity by replaying earlier recorded messages.

::.. PUF-Based Key Gen (1/4)

An indispensable premise for a large majority of cryptographic implementations is the ability to securely generate, store and retrieve keys. The minimal common requirements for secure key generation and storage are:

1. a source of randomness to ensure that freshly generated keys are unpredictable and unique, and
2. a protected memory which reliably stores the key's information while shielding it completely from unauthorized parties.

From an implementation perspective, both requisites are non-trivial to achieve. The need for unpredictable and unique randomness is typically filled by applying a (seeded pseudo-) random bit generator (PRNG). However, such generators are difficult to properly implement.

::.. PUF-Based Key Gen (2/4)

Implementing a protected memory is also a considerable design challenge, often leading to an increased implementation overhead and/or restricted application possibilities in order to enforce the physical security of the stored key.

A PUF-based key generator tries to tackle both requirements at once by using a PUF to harvest static but device-unique randomness and processing it into a cryptographic key.

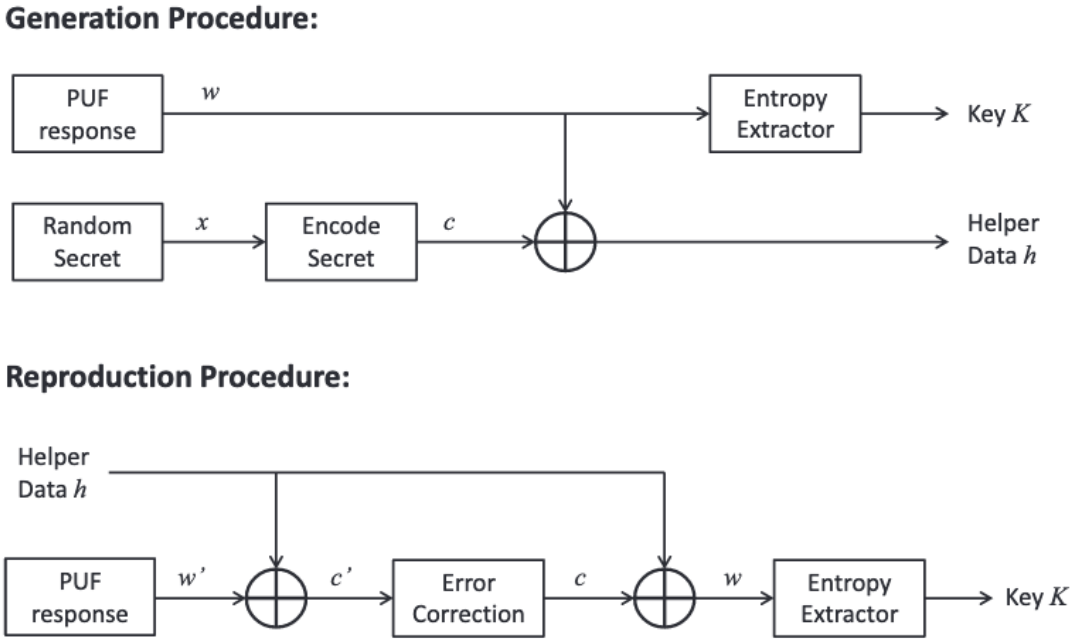
1. a PRNG is avoided, since randomness is already intrinsically present in the device is used,
2. No need for a protected nonvolatile memory, since the used randomness is considered static over the lifetime of the device and can be measured again and again to regenerate the same key from the otherwise illegible random features.

::.. PUF-Based Key Gen (3/4)

The min-entropy indicates how many bits of a PUF response are uniformly random. In order to be used in key generation, such a measure should be high; however, the minimum number of random bits observed in a response of the ring oscillator PUF, or other ones, is low, which means that some responses can be guessed with high probability.

A practical PUF-based key generation requires to abandon the requirement of generating information-theoretically random keys and aims for cryptographically secure randomness based on reasonable assumptions and best-practice techniques, aiming to gain in overall efficiency.

::.. PUF-Based Key Gen (4/4)

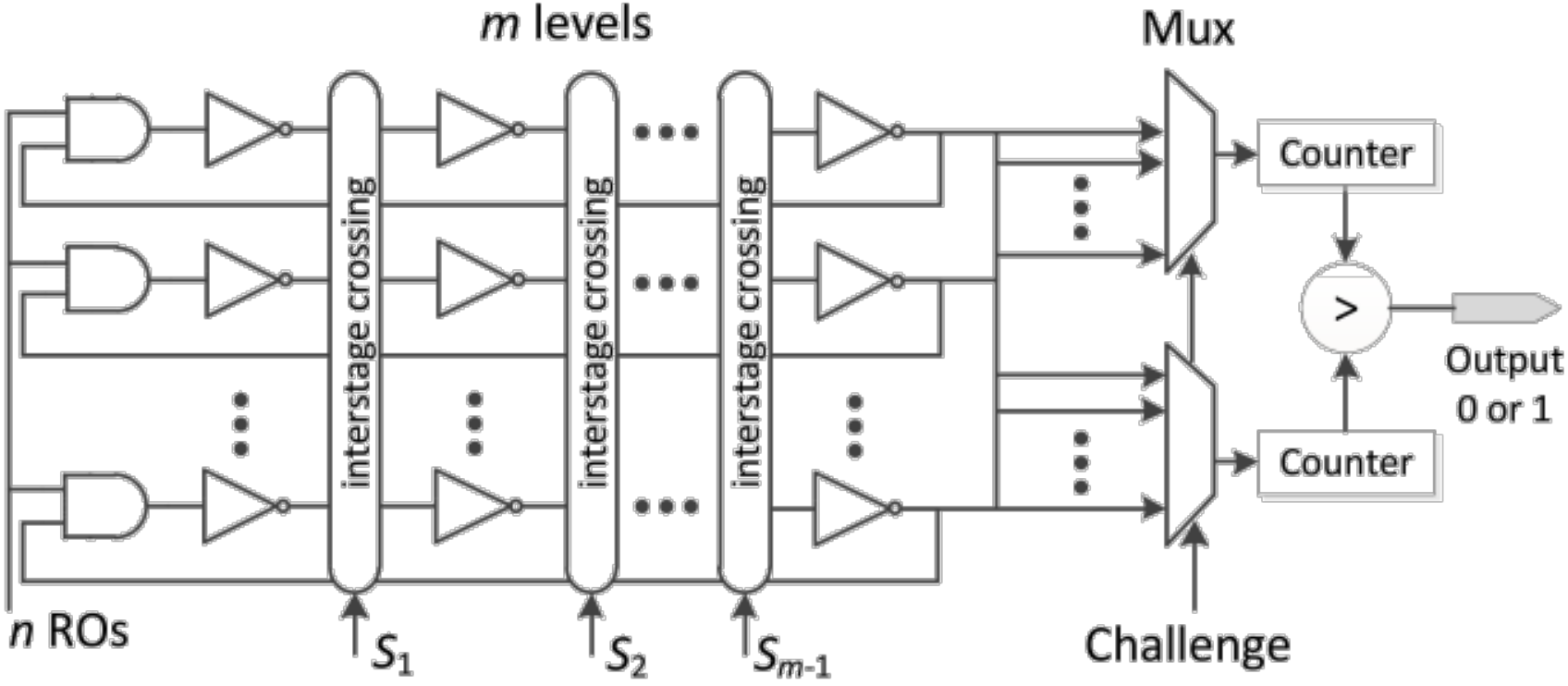


Generation procedure: Let C[n, k, d] be an error correcting code. On input $w \in \{0,1\}^n$, this procedure chooses $x \in \{0,1\}^k$ randomly and computes c=C(x), the encoding of x. The helper data is $h=w \oplus c$, while the the key K=H(w), where H is a hash function.

Reproduction procedure: On input of an element $w' \in \{0,1\}^n$, a helper data string h, and a universal hash function H, this procedure first computes $c'=w' \oplus h$. It then decodes c' to get c, if $dist(w,w') \leq t$. It reproduces $w=h \oplus c$ and outputs the key K=H(w).

::.. PUF-Based Key Sharing (1/2)

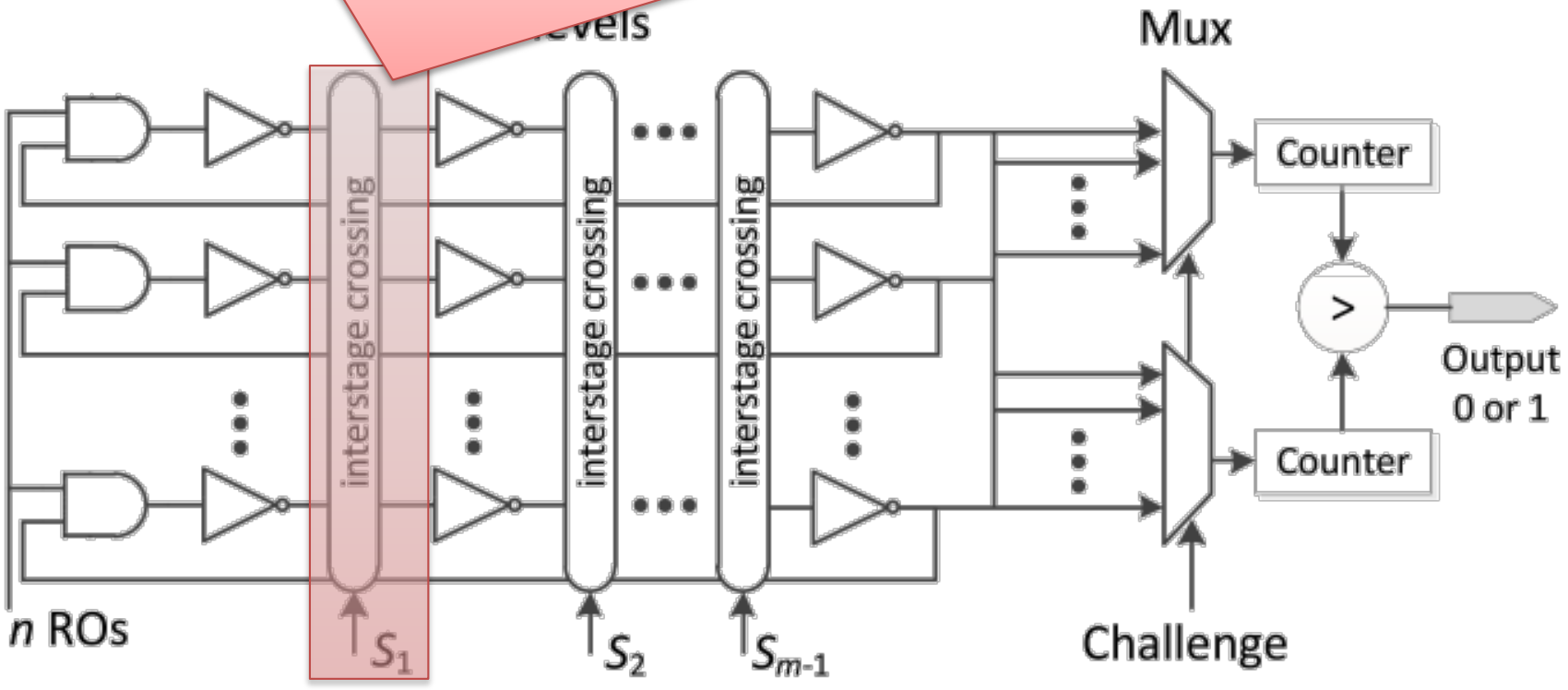
The shared key is required in multi-party communication between different devices, e.g., to establish a secure communication link. Traditional PUFs generate chip-unique key for every device, while CRO PUF is able to generate the same shared key for all devices.



::.. PUF-Based Key Sharing (1/2)

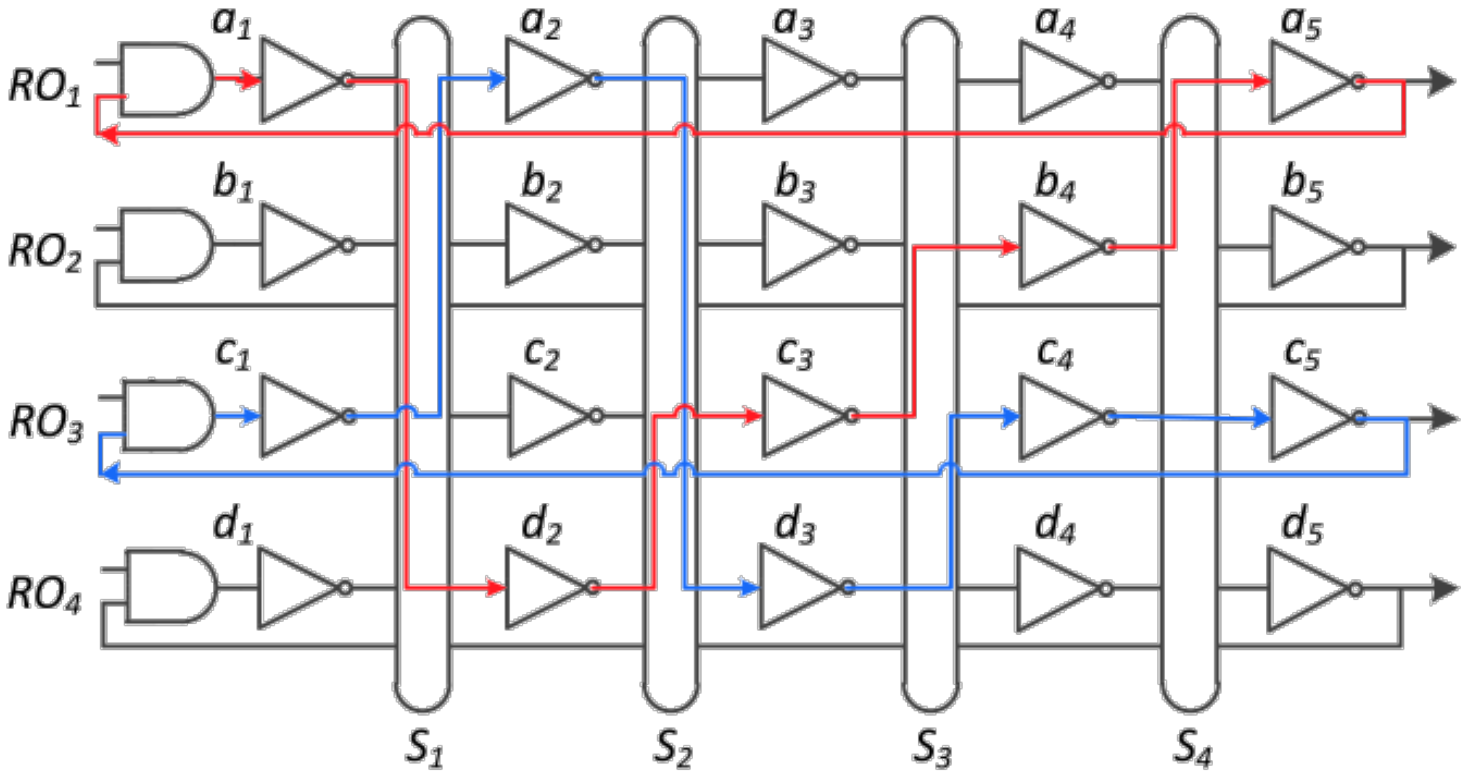
The shared key is required in multi-party communication between different devices, e.g., to establish a secure communication link. Traditional PUFs generate chip-unique key for ev
share

The interstage crossing cell determines the routing path of step signals input without any additional logical operation.



::.. PUF-Based Key Sharing (1/2)

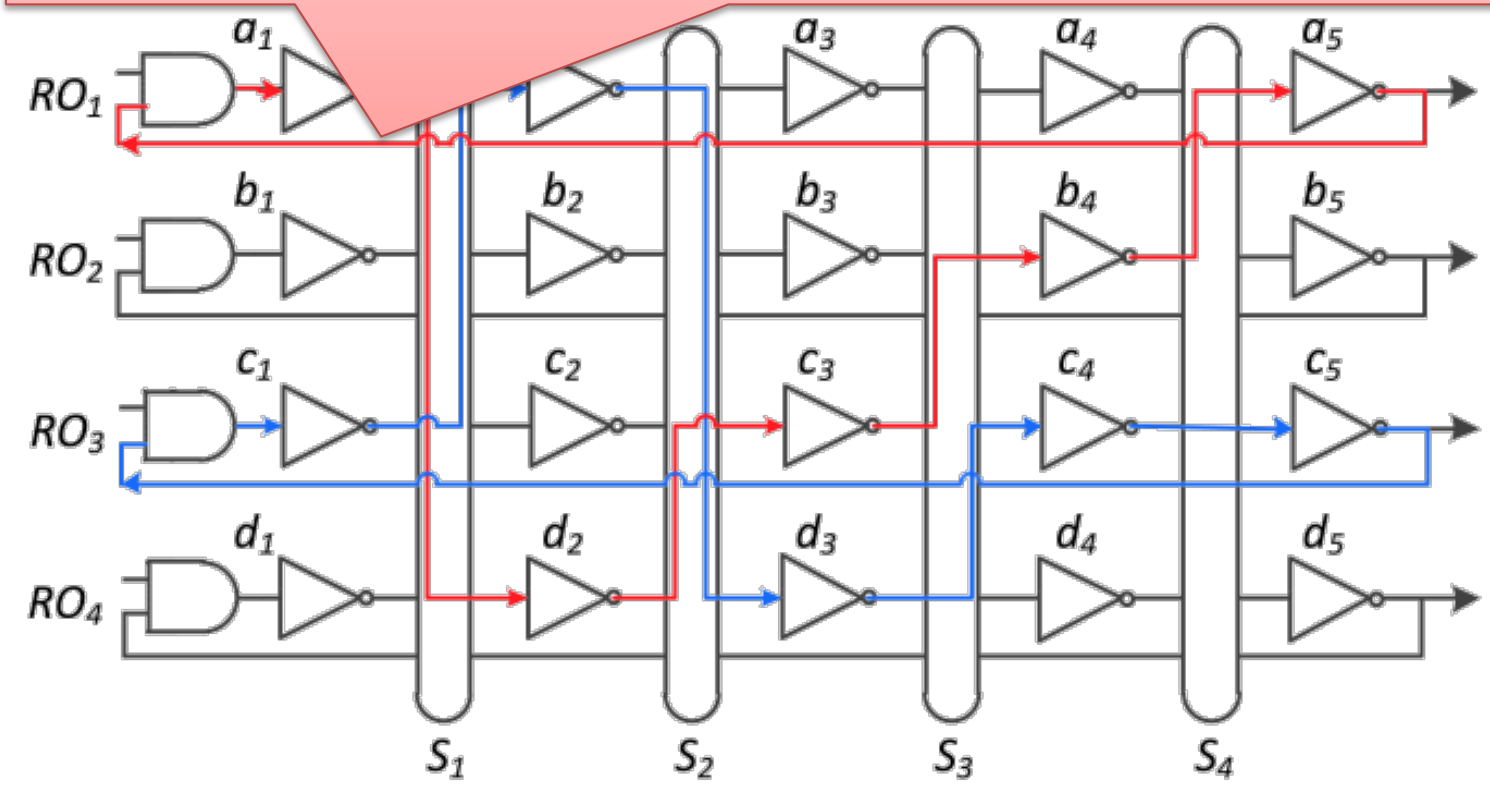
The shared key is required in multi-party communication between different devices, e.g., to establish a secure communication link. Traditional PUFs generate chip-unique key for every device, while CRO PUF is able to generate the same shared key for all devices.



::.. PUF-Based Key Sharing (1/2)

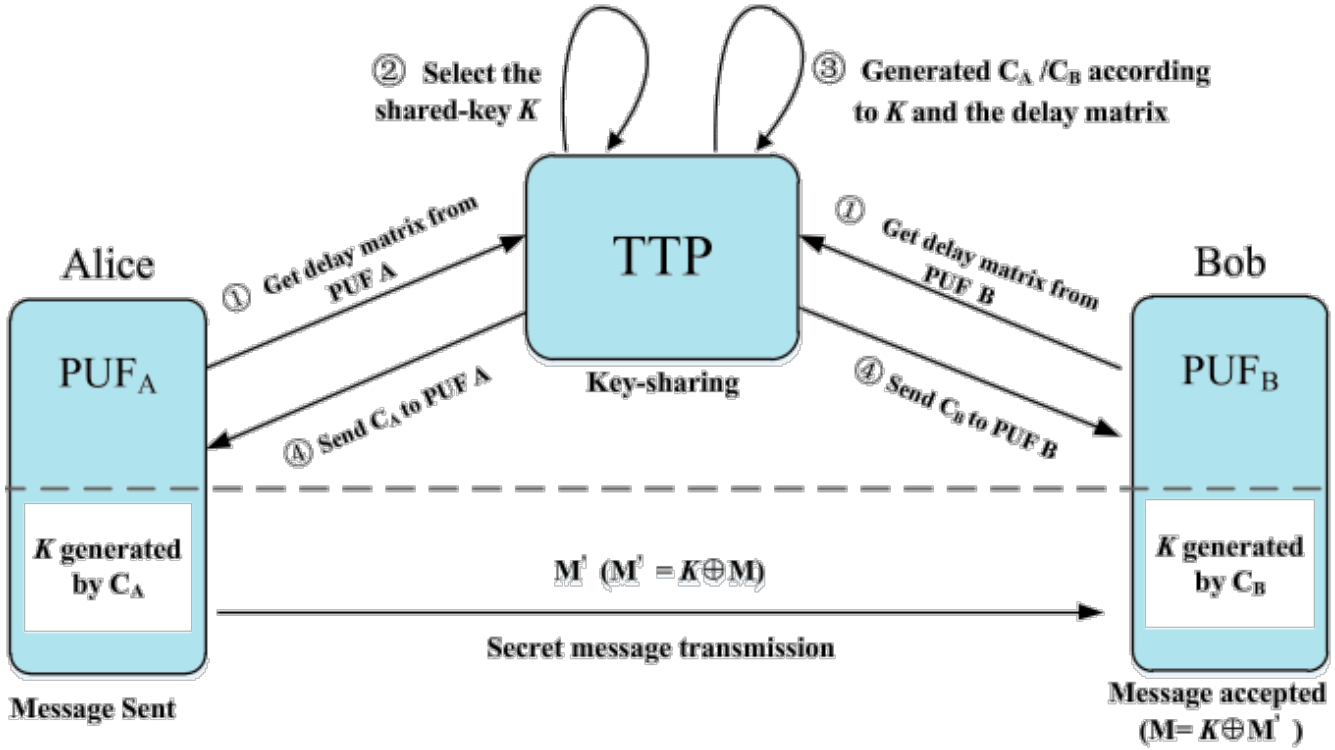
The
betw
comn
for e
share

The selection signal S adjusts the delay of each column with the function f. Challenge uses the function g to select different rows of ROs for comparison to generate the response. The delay is different between any two CRO PUFs, but we can get the same response by using function f and g



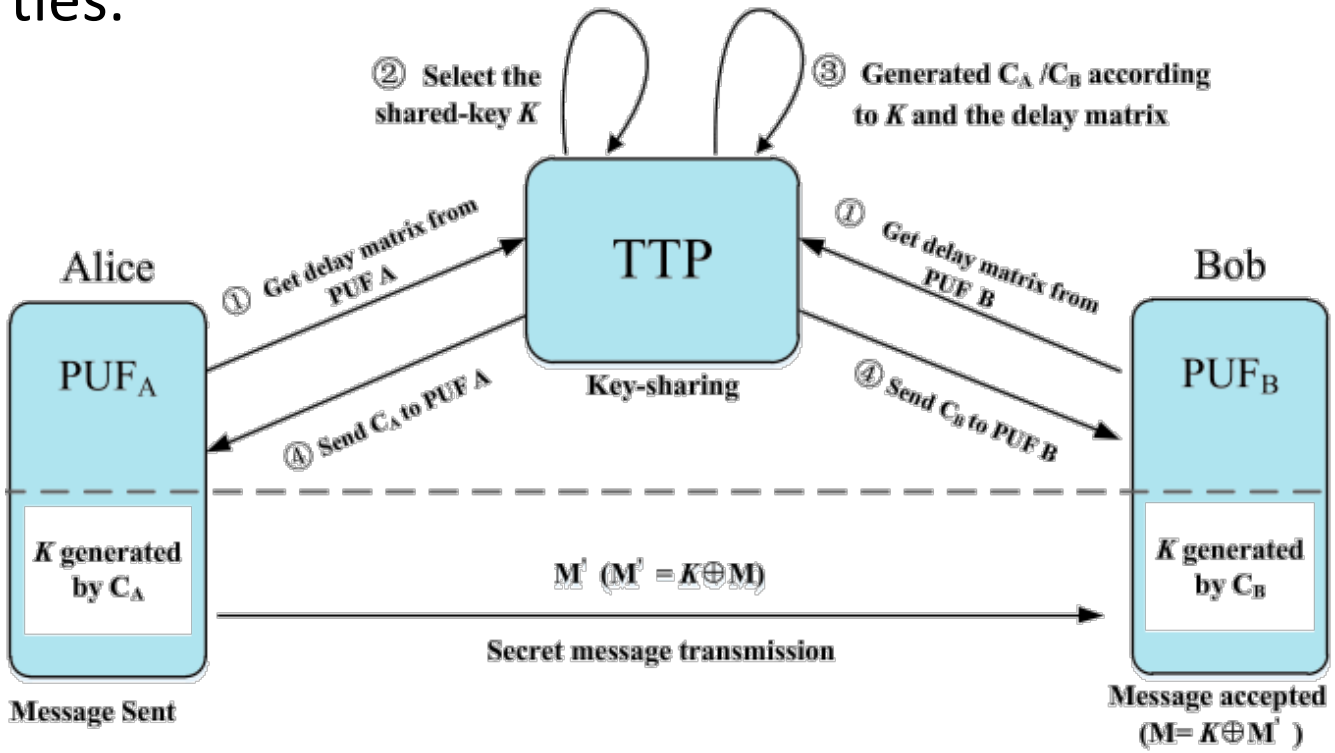
::.. PUF-Based Key Sharing (2/2)

First, we send the delay matrix of PUF_A and PUF_B to the TTP. Second, the TTP selects a key K that needs to be shared between PUF_A and PUF_B. Third, TTP generates the challenge C_A and C_B according to the delay matrix and the shared-key K. Then, TTP sends C_A and C_B to PUF_A and PUF_B, respectively.



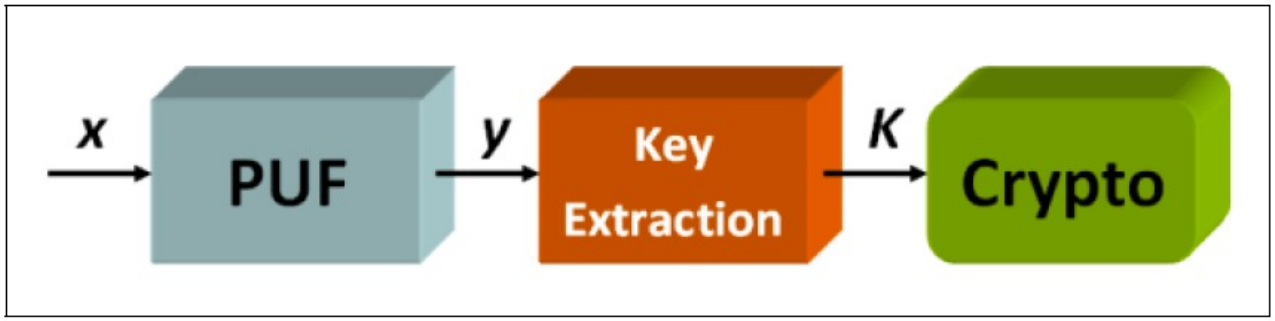
::.. PUF-Based Key Sharing (2/2)

Finally, PUF_A and PUF_B are able to generate the shared-key K with C_A and C_B, respectively. In the whole process, there is no secret key transmission. Besides, configuration information S is not the secret information and can be stored in SRAM. Therefore, it is with high security and low cost to realize the key-sharing among multi-parties.

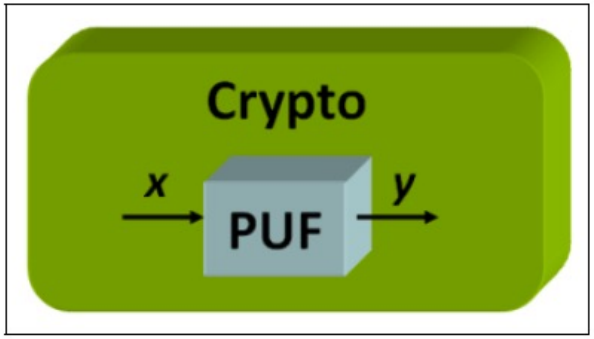


::.. Hardware Entangled Cryptography

In existing cryptographic primitives (a), secret key generations from PUFs are designed and addressed in the application scenarios. However, PUF can be also fully integrated in the crypto-primitive itself, which is termed as hardware entangled cryptographic primitives. These primitives are keyless since a secret key is not passed to the primitive or stored in either volatile or nonvolatile memory, so as to offer higher security against attackers.



(a) Classical cryptography with PUF-based secret key generation.



(b) Hardware entangled PUF-based cryptography.