Fakultät für Elektrotechnik und Informationstechnik
Technische Universität München

Metrics for Physical Unclonable Functions

Florian K. A. Wilde

Vollständiger Abdruck der von der Fakultät für Elektrotechnik und Informationstechnik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften (Dr.-Ing.)

genehmigten Dissertation.

Vorsitzende(r):
Prof. Dr.-Ing. Ralf Brederlow

Prüfer der Dissertation:
1. Prof. Dr.-Ing. Georg Sigl
2. Prof. Debdeep Mukhopadhyay, Ph. D.

Die Dissertation wurde am 10.06.2021 bei der Technischen Universität München eingereicht und durch die Fakultät für Elektrotechnik und Informationstechnik am 22.11.2021 angenommen.

Abstract

The concept and expected properties of PUFs are discussed w.r.t. the statistical and cryptographic concepts of random fields and random oracles. An intuitive classification in single, multi, and arbitrary challenge PUFs is introduced. The large variety of existing metrics to assess the reliability and unpredictability of PUFs is shown in a detailed overview. A common notation with the five dimensions accesses, response bit positions, challenges, devices, and environmental conditions makes comparison of the metrics easier. Experiments are performed on publicly available datasets by Su et al., Maiti et al., Hori et al., and Wilde.

Several common metrics are identified that contain the same information, e.g. Bit-Alias, Uniqueness, and inter-class bitwise entropy are aggregates of the mean along the axis of devices (MoD), while Reliability, bit error rate, Steadiness, Correctness and intra-class min-entropy are aggregates of the mean along the axis of accesses. Weaknesses in Randomness, mean Bit-Alias, and Uniqueness are found due to the way these metrics calculate the mean in multiple axes. A univariate and multivariate Bernoulli, and a multivariate categorical model are introduced to represent the statistical challenge-response behavior of ideal and of realistic PUFs. These models allow to find the expectation and sensitivity of Uniqueness and other aggregates of the MoD. Confidence intervals for the MoD are established based on those for the binomial proportion, and the achievable CI width for a selection of previous work is calculated. To evaluate the unpredictability of a PUF, compression with a modern context-mixing based compressor provides tighter bounds than with the usual CTW method. The result of TRNG test suites when used to assess PUFs is affected by the way multidimensional data is flattened, which requires to test with different flattening orders or adapt the tests to PUFs.

Statistical hypothesis tests are developed from Uniformity and Bit-Alias to simplify assessment. To assess spatial autocorrelation, principal component analysis and spatial autocorrelation analysis (SPACA) are introduced to the field of PUFs for binary and analog response data. Among the SPACA methods Moran's I, Geary's c, and join count, the last is implemented on an FPGA as part of a built-in self test. The response mass function (RMF) is introduced to represent the probability distribution of PUF responses even with thousands of bits under any of the three statistical models. The RMF finally allows to calculate the expected conditional min-entropy in a fuzzy commitment and fuzzy extraction scenario without the need for an IID assumption on the PUF output.

Zusammenfassung

Das Konzept und die erwarteten Eigenschaften von PUFs werden unter Berücksichtigung der aus Statistik und Kryptographie bekannten Zufallsfelder und Zufallsorakel diskutiert. Dabei wird eine intuitive Einteilung von PUFs in solche mit einfach-, mehrfach-, und beliebiger Anfragemöglichkeit eingeführt. Es folgt ein detaillierter Überblick über die große Vielfalt an Metriken zur Messung der Zuverlässigkeit und Unvorhersagbarkeit von PUFs. Dabei hilft eine einheitliche Notation mit den fünf Dimensionen Zugriffe, Antwortbitposition, Anfrage, Geräte und Umgebungsbedingungen beim Vergleich der Metriken. Praktische Beispiele basieren auf den vier öffentlich verfügbaren Datensätze von Su et al., Maiti et al., Hori et al., und Wilde.

Unter den üblichen Metriken werden einige identifiziert, die die selbe Information enthalten. So sind z.B. Bit-Alias, Uniqueness, und die inter-Klassen bitweise Entropie abgeleitete Metriken des Mittelwertes über Geräte, während Reliability, Bitfehlerrate, Steadiness, Correctness und die intra-Klassen bitweise Entropie auf dem Mittelwert über die Zugriffe basieren. Schwächen in Randomness, mittlerem Bit-Alias und Uniqueness ergeben sich durch die Art, wie Mittelwerte über mehrere Dimensionen dort gebildet werden. Zur Modellierung des statistischen Anfrage-Antwort Verhaltens von idealen und echten PUFs werden ein univariates und ein multivariates Bernoullimodell sowie ein multivariates kategorisches Modell eingeführt. Diese erlauben, den Erwartungswert und die Empfindlichkeit von Uniqueness und anderen auf dem Mittelwert über Geräte basierenden Metriken zu berechnen. Für diese Art Metriken werden Konfidenzintervalle, ausgehend von denen für die Binomialverteilung, erarbeitet und die erreichte Schätzgenauigkeit in ausgewählten bestehenden Arbeiten überprüft. Bei der Bestimmung der Unvorhersagbarkeit von PUFs mittels Kompression erreichen moderne sogenannte context-mixing basierende Verfahren engere obere Schranken als das bisher übliche CTW Verfahren. Werden TRNG Tests für PUFs benutzt, hängt deren Ergebnis von der Art ab, wie mehrdimensionale Daten serialisiert werden, weshalb entweder mehrere solche Arten überprüft oder die Tests auf PUFs angepasst werden müssen.

Um die Bewertung von PUFs zu vereinfachen, werden Uniformity und Bit-Alias zu statistischen Hypothesentests weiterentwickelt. Zur Detektion von räumlichen Abhängigkeiten werden die Hauptkomponentenanalyse (PCA) und Verfahren zur Analyse räumlicher Autokorrelation (SPACA) in den PUF Bereich eingeführt und auf binäre sowie reellwertige Daten angewendet. Von den SPACA Methoden nach

Moran, Geary und dem sogenannten Join Count wird für die letzte zudem eine FPGA Implementierung als Teil eines Selbsttests für PUFs vorgestellt. Eingeführt wird die Antworthäufigkeitsfunktion, um die Wahrscheinlichkeitsverteilung der PUF Antwort unter allen drei oben genannten Modellen auch bei Antwortlängen von mehreren tausend Bit darstellen zu können. Dies erlaubt schließlich, die zu erwartende bedingte min-Entropie in fuzzy commitment und fuzzy extraction Anwendungen ohne eine IID Annahme für die PUF Antwort zu berechnen.

Dedicated to the future,
where probability matters,
because the past cannot be changed.

About This Work

Calculations for this work were mostly performed in Python 3.6.9 with packages numpy 1.18.3, scipy 1.4.1, libpysal 4.2.2, scikit-learn 0.22.2.post1, esda 2.2.1, pandas 1.0.3, nistrng 1.2.0, and others. The source code for the calculations and examples is available online and most easily found by the unique permanent ID of this document: db0e67612cbaf33b687df4c15a198025aead3392 Compile date of this version: 2022-2-11

This document was written in LaTeX using the scrbook class with packages circuitikz 1.0.1, glossaries v4.35, biblatex v3.10, and others. The title pages are provided by the book class of the LaTeX4EI group, whose products are available from `https://gitlab.lrz.de/latex4ei/tum-templates/-/releases`. The document was compiled using pdfTeX 3.14159265-2.6-1.40.18 (TeX Live 2017/Debian).

References in bold indicate that I am one of the authors of the referenced publication.

Contents

1. Physical Unclonable Functions

1.1. Concept

The era of digitalization brought mankind something fundamentally novel: lossless duplication. From this point onward, it was possible to create absolutely identical copies that no one, human or else, could tell apart. Despite the immense potential of this scientific revolution, the possibility of lossless duplication is most evident in the cases where someone wants it to be *impossible*. In the decades since the introduction of digital music – slowly starting with the invention of the compact disc (CD) and peaking with the invention of MPEG-1 audio layer III (MP-3) and publicly accessible internet – the big entertainment companies have been fighting a crusade against the work of artists spreading online without increasing their profit.

Despite this progress, there remains an area where lossless duplication is unattainable: our physical world. Ever since the industrial revolution, standardization has been one of the key factors for scaling up productivity and reducing costs per item. This led to more and more precise tools that were able to produce more and more homogeneous items. Quickly, our bare eye was no longer sufficient to distinguish two screws or taper pins of the same type, but required tools such as a caliper gauge. This reduction of manufacturing variations progressed so far that today, mechanical production using electrical discharge machining (EDM) achieves an accuracy of less than 1 µm and manufacturers of integrated circuits (ICs) are capable of reliably producing structures as small as 7 nm, which is equivalent to as few as 64 silicon atoms. However, analysis tools improved accordingly and the most advanced of them today have a resolution of 50 pm [1], 140 times smaller than a 7 nm structure. So no matter how small the manufacturing variations might become, they will most probably remain measurable with an appropriate instrument and infeasible to avoid entirely.

The idea behind physical unclonable functions (PUFs) is to use these inevitable manufacturing variations as a feature rather than a flaw. They provide unobtrusive distinguishability to otherwise identical objects, which can be desired for inventory keeping, forensics, or security in general. This idea is not entirely new; previous examples include human fingerprints and rifle round scratches left by the barrel. While fingerprints come in a few coarse types, such as whorl, loop, and arch, even nature is not able to reproduce these patterns perfectly. Instead, every fingertip has a randomly distributed set of ridge endings, bifurcations, very short ridges, etc., which altogether make it unique. Similarly, rifle barrels do not possess a perfectly smooth surface, which in itself is a feature used to provide spin to the round to stabilize its trajectory. Since the exact position of peaks and grooves in the barrel is random, though, the scratches on the round caused by them allows to prove which round came out of which barrel.

Note that the concept of PUFs is based on the problem of elimination of manufacturing variations, not on the problem of production itself. The latter problem has been used for a very long time already to make e.g. coins and bank notes unforgeable. They are equipped with hard to manufacture features, such as holograms, fine print art or highly detailed embossments, so only a few trusted facilities are able to produce these features at all. In contrast, the production of a PUF itself can be trivial without security impact, it should only be hard to produce an exact replication of some other PUF instance. This advantage opens up many new applications, where the use of a hologram would pose a prohibitive increase in production cost. The downside is that the mere fact that something bears a PUF is no longer a proof of authenticity. Instead the PUF has to be compared to a trusted list of authentic PUF instances.

An intuitive example is the optical PUF developed at the Massachusetts Institute of Technology (MIT) in 2001 by Pappu [2] – called physical one-way function (POWF) back then, but often referred to as the corner stone of PUFs. It uses coin-sized slabs cast from resin with small reflective particles mixed in. While the production is cheap and trivial, the size of the slabs and the ratio of reflective particles was chosen to cause coherent multiple scattering of an incident laser beam, so if lit up from a certain angle on one side, a complex and most likely unique speckle pattern is projected onto an adjacent screen, or camera, on the other side. To deliberately produce another token that produces the same speckle pattern, however, is quite the opposite of cheap, because it would require to first of all characterize the size, position, and orientation of the reflective particles with high, at best prohibitively high, accuracy and then place identically shaped particles within some resin with equally high accuracy. To just record the speckle pattern instead of characterizing the reflective particles in the token would be insufficient, because the pattern also depends strongly on the incident angle of the laser, so there is a very large number of angles for which a speckle pattern would have to be recorded. Additionally, the use of resin allows to cast the slab in a cavity of the object to be authenticated, so removal of the slab – for example to transfer it to a counterfeit object – would most likely cause damage to the resin and change the speckle pattern so it no longer passes an authentication process that expects the original patterns.

Pappu's optical PUF was not the first practical concept to use manufacturing variations as a feature rather than a flaw, though. A year before Pappu, Keith Lofstrom[1] proposed an *identification* circuit for ICs [5]. It did not feature a challenge input, thus was less suited for direct *authentication* of physical objects, because an attacker could simply eavesdrop on the response and send it from a counterfeit device. However, it would have been well suited for today's main application for PUFs, which is secure key storage for cryptography on ICs, cf. Sec. 1.3. Improved into a static random-access memory (SRAM)-like PUF design by Su et al. [6], it found its way into mainstream PUF publications, which emerged from Pappu's POWFs being adapted to ICs as "silicon physical random functions" by Gassend et al., where the name and abbreviation PUF was mainly chosen to avoid an abbreviation clash with pseudo random functions (PRFs) [7].

1.2. Discussion of Properties

The varying terminology until the term PUF emerged reflects the difficulty to capture the essential properties of the concept in existing mathematical language. This section therefore aims to work out a consistent set of properties that are nowadays to be expected from a PUF, and points out where they agree with existing mathematical objects or concepts, such as *random functions* (respectively *random fields*) and *random oracles*.

Random function is a term from probability theory and commonly used as a synonym for *random process*, which "is a mapping of outcomes of an experiment to functions of time" [8]. Time can be continuous such as during the observation of ambient temperature at a particular place on earth within a year, or can be discrete, such as by observation of a sequence of dice rolls. As the examples illustrate, a random function *may* assign independent and identically distributed (IID) values to each point in time as in the sequence of dice rolls, but not necessarily so as with temperature, which is a state variable that changes gradually, not instantly. A particular observation of a random function produces a so-called *sample function* [8], which is therefore a deterministic function. The generalization of a random process to multidimensional arguments is called a *random field*.

In the field of cryptography, a *random oracle* describes an oracle that responds to any query with a uniformly at random chosen answer from its predefined set of possible answers as long as the query is asked for the first time. If a query is repeated at some later point in time, the oracle will return the same answer as the first time the query was asked. Since the number of possible queries is infinite, it is infeasible to build in pratice. Instead, it is a theoretical concept to represent an ideal cryptographic hash function. If the infinite set of possible queries is

[1] An electrical engineer who is better known for his "launch loop" design [3] to launch objects into space without rocket propulsion, which has been featured in several works of science fiction [4].

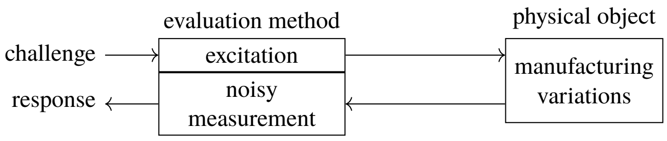


Figure 1.1.: Interaction of evaluation method with physical object during an access to a PUF.

represented by the infinite set of natural numbers, random oracles correspond to random functions that assign independent and uniformly at random chosen values to each argument. Since the randomness in the choice of response can not be observed from repeated queries to the same random oracle, a *particular* random oracle corresponds to a sample function.

1.2.1. Being a Function

Based on their origin in POWFS, PUFS are generally considered to be functions in the mathematical sense, i.e. a mapping of values from one set to another set. This property is shared with both random functions and random oracles. It is realized by an evaluation method, which is highly specialized to the physical object of which the manufacturing variations should be utilized. Pappu's optical PUF [2] makes this particularly easy to grasp: The physical object is the slab made from resin with reflective particles, while the evaluation method is to use a laser, camera, and fixture to measure speckle patterns. With electronic silicon PUFS that are part of an IC, the distinction between both may be less obvious, but still applicable, see for example [9].

The domain, in the PUF context called challenge space $\mathcal{W}$, encompasses permissible parameters for how the evaluation method should interact with the physical object and what constitutes the output. The latter is commonly referred to as response, and consequentially the codomain is called response space $\mathcal{X}$.

$$\mathrm{PUF} : \mathcal{W} \to \mathcal{X} \tag{1.1}$$

The evaluation method is hence a physical implementation of a function that takes into account the manufacturing variations of the physical object. This process is illustrated in Fig. 1.1.

While the physical world is inherently analog, as are the stimuli and responses to the physical object itself, it is common in PUF literature to consider mostly their binary representation. It helps to efficiently store challenge-response pairs (CRPS) and recognize a previous response based on the number of equal bits. Therefore

$$\underline{w} \in \mathcal{W} \subset \{0,1\}^*, \tag{1.2}$$
$$\underline{x} \in \mathcal{X} \subset \{0,1\}^*, \tag{1.3}$$

where $\{0,1\}^*$ is the set of all possible binary vectors. The term *space* is often used informally in the field of PUFs, but it does hold in the mathematical sense, too, since both spaces are usually expected to feature some kind of *distance* between their elements, which makes them *metric spaces*. The most common distance metric is the Hamming-distance (HD), which is equivalent to the Manhattan distance for subsets of $\{0,1\}^*$.

Based on the terminology of random functions,

$$\mathbf{PUF}(\underline{w}) \tag{1.4}$$

denotes the random field of some PUF design and with $\underline{o}_d$ the binary description of the manufacturing variations in a particular instance of this design on a device d,

$$\mathbf{PUF}(\underline{w}, \underline{o}_d) \tag{1.5}$$

or more briefly

$$\mathbf{PUF}_d(\underline{w}) \tag{1.6}$$

is the corresponding *sample function*, i.e. a deterministic function with fixed mapping based on a particular observation of the random variable (RV) or RVs described by $\underline{o}_d$. Using the analogy to random oracles, one may either consider the design a single random oracle that is provided queries in the form of (1.5) or every instance an individual random oracle that is provided queries in the form of (1.6). Both approaches yield the same behavior.

The cardinality of the challenge space is typically finite for PUFs and fixed for a particular design, whereas random oracles and one-way hash functions (OWHFs) map infinite challenge spaces to finite response spaces. From this point of view, Pappu's name choice of POWFs over physical one-way hash functions (POWHFs) fits, but either name highlights one-wayness although this is no important property[2] of PUFs for most applications. An infinitely large challenge space is not necessary for a PUF, though, because a finite domain may still be infeasible to enumerate in practice, which can be relevant to achieve the property of unclonability, cf. Sec. 1.2.4. However, PUFs with feasible to enumerate challenge space can be useful for applications that use the CRPs internally, if additional protections ensure that

[2]A fact also mentioned e.g. in [9].

the CRPs cannot be extracted by an attacker. Even PUFs with unary challenge space, such as IC identification circuits and SRAM PUFs, are practically relevant. In this edge case, the mere act of applying power to the PUF constitutes the only possible challenge, which means that a PUF does not necessarily comprise a challenge input, except for, continuing the line of thought, its power input. This fits well for the use in ICs, because it means that the CRP can only be extracted when power is applied, and it is thus sufficient to have the protections against read-out active during this time.

Similar to the challenge space, the cardinality of the response space is not necessarily large. Several popular PUF circuits, cf. Sec. 1.5, provide only a single bit of output. The entropy required by the application is then usually established by concatenation of responses from repeated evaluations for multiple different challenges, from multiple instances placed next to each other, or a combination of both. Note that if multiple instances are placed next to each other, they may inadvertently interfere with each other, so it is reasonable to consider the whole structure a PUF comprised out of *cells*. In this case the response space can reach large cardinality, e.g. 2^{256} for 256 cells if each cell contributes one bit to the overall response.

1.2.2. Easy to Evaluate

To be of practical value, the evaluation of the function, i.e. a query to a PUF, should not be a Herculean task. According to [2], it should be in the complexity class $\mathcal{O}(1)$, i.e. constant time. However, constant can still be very long and it seems reasonable that a more complex physical object or more sophisticated evaluation process takes longer, but may also provide more useful information. Therefore the definition from [7], which requires the evaluation to take "a short amount of time"[3], should be extended to "a small amount of resources". It then also captures e.g. energy consumption during the evaluation process, required die space, etc. What exactly is a small amount of resources is defined in [7] as "linear or low-degree polynomial" in the size of the device, which is a security parameter there. While the size of the physical object was related to its complexity for the types of PUF known back then, in particular the optical PUF [2], the relationship was mixed up with the development of mostly silicon based PUFs, which provided much more complexity per unit volume. Therefore the "small amount of resources" should rather be related to the amount of information the response carries, e.g. measured by information theoretic entropy. However, it should also be related to other properties such as reliability, unpredictability, or unclonability, because an

[3]Note that Gassend et al. did write about resources in the context of the "hard to characterize" [7] property, but not of the "easy to evaluate" [7] property, where they only considered time.

improvement in one of these properties may justify to spend additional resources, e.g. a longer time to response for higher reliability, or additional die space for higher entropy per response bit. Since the priority of properties may differ by application, performance metrics should preferably report required resources such as time or energy individually to allow application dependent choices.

1.2.3. Reliable

In contrast to a true random number generator (TRNG), a PUF is expected to provide the same response every time the same challenge is applied to the same instance, just as a random oracle produces the same response if the same query is repeated. In computer science terms, a TRNG provides run-time randomness, while the purpose of a PUF is to provide compile-time, or rather manufacturing-time randomness. However, as PUFs are physical objects, they are inherently exposed to the effects of aging, temperature, supply voltage, etc. Furthermore, they show a certain amount of run-time noise, because PUFs utilize manufacturing variations that are often relatively small. While a design goal for PUFs is to minimize the change in response due to run-time noise and environmental conditions, the probability of an incorrect response remains non-negligible today and post-processing of the response e.g. through error correction codes (ECCs) is common. The mathematical notation is therefore extended by introduction of index e that reflects the operating conditions of the PUF, and index a that denotes a particular access at a particular point in time. An ideal PUF then satisfies

$$\mathbf{PUF}_{a,d,e}(\underline{w}) = \mathbf{PUF}_{a',d,e'}(\underline{w}) \quad \forall a, a', d, e, e', \underline{w}. \tag{1.7}$$

An equivalent statement based on the corresponding responses $\underline{x}_{a,c,d,e}$ and $\underline{x}_{a',c,d,e'}$, with c as index for a particular challenge, is

$$\underline{x}_{a,c,d,e} = \underline{x}_{a',c,d,e'} \quad \forall a, a', c, d, e, e'. \tag{1.8}$$

For non-ideal PUFs to be useful, the effects of run-time noise and environmental conditions have to be either negligible or able to be compensated. While compensation may be possible for certain deterministic effects of environmental conditions such as temperature, run-time noise adds another random process on top of the manufacturing variations. Thus the mapping embodied by the PUF is not so much between a challenge and a response, but between a challenge and a certain probability distribution on the response space. The expected behavior of a real-world PUF therefore is to provide a sufficiently similar response with high probability, i.e.

$$\mathrm{P}\left(\|\bar{\underline{x}}_{c,d}, \underline{x}_{a,c,d,e}\| \leq \epsilon\right) \approx 1 \quad \forall a, c, d, e, \tag{1.9}$$

where $\| \cdot \|$ denotes a suitable distance metric such as HD, ϵ is an acceptable amount of error due to environmental conditions and run-time noise, and $\underline{\bar{x}}_{c,d}$ is the so-called *true response* for challenge c on device d. To satisfy this, the run-time noise process needs to have decent properties, i.e. its expectation exists and changes only negligible over time, and its variance is sufficiently small at virtually all times.

For the definition of true responses, two ways are common that differ in the amount of expected error. First, it may be obtained from the mean of a sufficient number of accesses under reference environmental conditions, in which case it approaches the expected response, i.e. the expectation of the run-time noise superimposed on the theoretical response without noise. This minimizes the expected error through run-time noise to its standard deviation. Second, an arbitrary access at reference environmental conditions may be chosen as true response. This increases the expected error depending on how far the selected response happens to be off of the expected response, but since the variance of the noise process is assumed sufficiently small, it may be an acceptable trade-off against the effort to average multiple accesses. Furthermore, since reference conditions need not be lab conditions, it may be calculated from measurements under multiple environmental conditions in a way that minimizes the expected error due to run-time noise over the entire range of permissible environmental conditions [10].

So reliability refers to the ability to obtain a sufficiently similar response with high probability, if the same device is queried with the same challenge despite different environmental conditions and run-time noise. Performance metrics for this property may be based e.g. on the probability of an incorrect response, the expected number of incorrect bits in a response, the amount of error correction needed to achieve a given probability of a correct ECC output, etc.

1.2.4. Unclonable

The most notable property, based on the name PUF, is unclonability. In a broad sense, it is commonly interpreted as infeasibility, given only a reasonable amount of resources, to produce an equivalent of a PUF that would make access to the original PUF instance unnecessary. A necessary but not sufficient prerequisite for this is that the function is not a deliberate implementation of some mathematical description, but the result of complex physical processes, thus a *physical function*. This relates once more to random oracles, which are black-box functions, i.e. functions whose internal workings are unknown. But even without knowledge of the internal workings, it may be possible to imitate the PUF's challenge-response behavior (CRB) through a cleverly designed model, and this model may be implemented in an IC together with appropriate input and output circuitry to make it indistinguishable from the outside. Unclonability thus incorporates two things: First, the inability to manufacture a *replica* of the physical object; and second, the inability to build a machine that *imitates* the physical object in some, many, or nearly all characteristics.

Previous attempts to make such a differentiation included the notions of mathematical unclonability [9] or logical unclonability [11], compared to physical unclonability. The motivation to differentiate for Maes et al. [11] was to claim physical unclonability for their PUF design although it was feasible to record a full set of CRPs, which, according to them, means their design is not logically unclonable. Armknecht et al. [9] made the differentiation to represent that the CRB of PUF candidate designs for ICs have repeatedly been imitated by mathematical models obtained through machine learning (ML) e.g. [7, 12], [13], while it remained infeasible at that time to produce a replica for any of the designs. With the continued success of ML based models the term physical*ly* unclonable function, which appeared already in [9, 14], [15], became more widely used. As its first word is an adverb to the second rather than an adjective to the third, it seems to reflect that physical unclonability is what can be expected from or what matters for a PUF. This change, however, does not consider the full picture. First, the adjective *physical* is a necessary specification of *function*, as mentioned in Sec. 1.2.1, to indicate that a PUF is not an implementation of some mathematical formula, but the result of complex physical processes. Otherwise any mathematical engine capable of evaluating the formula would be an imitation and there would be no physical object to replicate. Second, it remains unclear whether physical unclonability only refers to the inability to manufacture a replica or includes physical implementations of mathematical clones, too. Depending on the complexity of the model and the progress in Moore's Law, an application-specific integrated circuit (ASIC) implementation of a mathematical model might even be of the same size, require the same amount of energy and time as the original PUF design, which means it is able to imitate nearly all characteristics and is therefore hard to detect. To differentiate between replication and imitation is thus more obvious.

Another aspect of unclonability where multiple different definitions exist are the means by which an equivalent may be obtained and the amount of characteristics that an imitation has to resemble. The remainder of this subsection therefore outlines some common notions of unclonability, where an ideal PUF fulfills all of them, but even the weakest notion might justify the use of the term PUF.

Accidental Replication

The weakest notion of unclonability refers to the risk that, while the PUF design is used in mass production without malicious intent, it is sufficiently difficult for an attacker to obtain either a device that bears a replica of the physical object on another device to be attacked or any two devices that bear replica of each other. Previous work, e.g. [9], refers to this as the honest manufacturer scenario and distinguishes the two cases as selective and existential replication respectively. The notion merely implies a sufficient amount of manufacturing variations within the physical object, of which a sufficiently large part is captured and reflected in the CRB by the evaluation process. The latter requires, among others, to scale challenge space, response space, or both according to production volume to ensure the manufacturing variations can be represented in the CRB.

Accidental replication is the notion of unclonability typically chosen for PUFs intended to serve as a measure of IC identification, i.e. as an intrinsic unique identifier to replace externally produced serial numbers that need to be programmed into an IC or imprinted on the surface. It is unsuited if the PUF is to be used for security purposes, because it does not consider intentional replication or imitation.

Intentional Replication

A more stringent notion of unclonability allows an attacker to tweak the production process or use other, possibly more advanced manufacturing techniques to produce a replica. Still, the notion does not consider imitation, but requires to come up with a replica of the physical object. It thus particularly applies to scenarios where the evaluation is performed by an external reader that is separate from the device that carries the physical object, e.g. a credit card as described in [2, 16]. If such a PUF equipped credit card is to be used in e.g. a retail store or hotel, the employee at the counter would notice if the PUF had been replaced by an imitation that looks considerably different, is much heavier, or has a battery attached. It also applies to ICs, though imitation may be easier to achieve and sufficient there.

A PUF that aims to be unclonable in the notion of intentional replication therefore requires that it is sufficiently difficult:

1. either to measure a sufficiently accurate description of the physical object including its manufacturing variations,
2. or to reproduce the description in a physical object with sufficient precision.

If unclonability is based on the former, it requires clever design of the evaluation method, which has to be able to reliably extract CRPs without revealing an accurate description of the physical object. It furthermore requires that there is no other measurement technology that is able to extract such a description with reasonable effort, such as a side-channel attack (SCA) or fault attack (FA) during normal evaluation. To protect against such investigation, it has been argued, e.g. by Guajardo [17], that PUFs are inherently tamper evident and thus any such investigation would change the physical object in a way that substantially changes CRB and thus prohibits obtaining an accurate description. This argument neglects two things, though: First, many measurement techniques are optimized to have only minimal effect on the object to be measured, so investigation does not necessarily imply a sufficient modification to change CRB. As an example, Pappu demonstrated for an optical PUF that a 1 mm hole, although shallow, changes the response to a certain challenge noticeably [2], but this is a much stronger modification than e.g. an X-ray computed tomography (CT) scan would induce. Similarly, SCAs on electronic silicon PUFs do not change the CRB in any detectable way, and the change in CRB induced by removal of the IC's packaging is about as little as usual readout noise [18]–[20]. Second, even a complete destruction of the physical object, e.g. through iterative delayering combined with scanning electron microscope (SEM) scans of the layers, can be acceptable, if the obtained description allows to produce one or more replica of the physical object.

The latter option to protect against intentional replication is taken e.g. by Pappu [2] and requires that the manufacturing process is already optimized so that the remaining manufacturing variations are hard to control and hard to reduce further. Additionally, no other manufacturing technique of reasonable effort may exist to reproduce the description in a physical object with sufficient precision. A prerequisite for this is that measurement technology is always ahead of manufacturing technology, as outlined in Sec. 1.1, because, once again, the intended evaluation method must be able to reliably extract CRPs. Note that the evaluation method is also important in this approach, although a replication of the physical object is requested. If, for example, the evaluation method checks a certain parameter only to be larger than some limit, the manufacturing process or description can be tweaked to make this parameter so much larger that it fulfills the condition despite its own variations. Considering all manufacturing variations are rather small, a physical object produced this way may still count as a replica, because it is hard to distinguish from the original.

A special case in this regard are so-called protective PUFs, such as the coating PUF [21] or the PUF foil envelope [22]. Their physical design is aimed to insulate the evaluation circuit from an attacker by the physical object, so the latter would need to be penetrated thus significantly modified to measure the physical object's properties from the same perspective as the evaluation circuit, which makes this approach pointless, of course. The benefit of this approach is that the evaluation circuit is in a superior position to measure the physical object's manufacturing variations, so an attacker would require a more sophisticated measurement tool to compensate for the disadvantage. A further benefit of protective PUFs is that additional circuitry can be placed on the inside, which is then likewise protected against access or manipulation by an attacker.

To predict advancements in measurement and manufacturing technology can be difficult, especially over long periods of time. A PUF should therefore contain sufficient margin regarding measurement or production accuracy to remain unclonable for its intended service time. For the same reason, the technology used to produce the PUF has to be updated regularly to keep pace with advancements in this area.

Imitation of the CRB

The most difficult to achieve notion of unclonability is the one where an attacker only needs to resemble the CRB, optionally together with a limited number of additional properties, e.g. time-to-response. It comes in many variations, such as whether the attacker has physical access to the original PUF for some limited amount of time, can read out arbitrary CRPs, or only eavesdrop on CRPs. The notion suits applications such as remote authentication or key storage.

The most prominent way to imitate the CRB of electronic silicon PUFs have been for almost two decades now ML based models. After training with a small portion of CRPs, which have either been eavesdropped during legitimate authentication runs or read out in feasible time from the PUF to be imitated, they are often able to predict the remaining CRPs with very high accuracy, e.g. 97 % [23], 99.9 % [13], 99 % [24], 98–99 % [25]. The resulting error of such models may even be lower than the run-time noise of the original PUF, so artificial noise may be added to the output of the models to avoid that the imitation is detected. At the same time, the necessary size of training data is so small that recording very few, sometimes even just one, authentication run is sufficient. Training times are correspondingly in the range of seconds to minutes for data measured from real hardware (HW) [24]. Improvements to the PUF circuits against this kind of model building attacks were repeatedly overcome by improvements to the model building techniques and increase in computational power, so training time remained in the range of hours [13]. The success of this type of attack rests upon failure to achieve another property of PUFs, which is unpredictability, cf. Sec. 1.2.5. Note that because this notion of unclonability only requires to resemble the CRB, it does not matter if the model requires a full-fledged workstation as long as it is within the amount of reasonable resources given to attack the PUF.

An alternative approach, e.g. for cases where CRPs are not available or ML on them does not produce a satisfying model, is to measure an accurate description of the physical object's manufacturing variations and simulate the lower level physical processes relevant to the CRB. To protect against this, either an accurate measurement or this kind of simulation has to be infeasible with the given resources. For example, coherent multiple scattering of light was chosen in [2] and the analysis in [16] showed that at most ten scatter events could be simulated with reasonable effort, much less than occur in the physical object. In contrast, at least some types of electronic silicon PUF can be simulated on an electronic level using standard SPICE based Monte-Carlo-simulations from IC design tools, see e.g. Rahman et al. [26] and Kalyanaraman et al. [27], who propose novel ring-oscillator (RO) respectively metal-oxide-semiconductor field-effect transistor (MOSFET) subthreshold current based PUF candidate circuits purely on simulated data.

An intermediate way to the previous approaches is to imitate the CRB based on a high-level description of the manufacturing variations, which may for example be obtained through SCAs. This includes the frequencies of an RO PUF or which transistors of an SRAM PUF are conducting [28]. Inference of the CRB based on this description can be trivial.

Finally, an attacker may simply build a look-up table (LUT) of all CRPs. To protect against this, the challenge space needs to be sufficiently large so it is infeasible with the given resources to record a more than negligible portion of CRPs. For example, one of the early optical PUF designs intended to authenticate smart cards would have required hundreds of days to record all CRPs, which was considered too long to go unnoticed [16], thus infeasible with given resources. Note that to avoid the pathological case where an attacker must record only one response if the PUF has unary challenge space, cf. Sec. 1.2.1, the response of such PUFs must only be used internally, so it can be assumed inaccessible for an attacker. What such internal use may look like is shown in Sec. 1.3.

Additionally and in particular if the device to be imitated is not accessible, the imitation may be improved based on the CRB of a reasonable amount of similar devices, such as those of the same type and from the same manufacturer. The success of this again depends on the unpredictability of the PUF candidate design.

Imitation of Nearly All Characteristics

Unclonability with regard to imitation can be easier to achieve if the imitation is required to resemble nearly all characteristics of the original PUF, such as size, power consumption, etc. It applies to cases where the PUF is used locally, but embedded into a device such as an IC. In this case an imitation would have to fit into the same die space or unused space in the same package and must not attract attention due to e.g. higher power consumption. Yet it does not require a replica, because the package hides the fact that it is an imitation rather than a replica.

From an attacker point of view, this increases the required resources, because the LUT or model must be cast into some piece of HW, which may exceed the given amount of resources to attack the PUF. It can even render it infeasible, e.g. because the model requires more computational power than available die space and power consumption permit. There are cases, though, where it is easy to imitate all characteristics: Once the response of a PUF with unary challenge space is measured through a SCA or predicted based on the response of other devices, it is trivial to replace the PUF by some non-volatile memory (NVM) programmed with the correct response and optionally add a delay to resemble time-to-response and waste some power to consume the same amount of energy. A large cardinality of challenge space does not necessarily prohibit such implementations either, see e.g. the highly accurate yet comparatively simple models for arbiter PUFs from [13, 24], which seem trivial enough to replace the actual arbiter PUF in an ASIC without significant increase in die space or power consumption.

1.2.5. Unpredictable

The last property of PUFs is unpredictability, which means that the response to each unique query appears to be drawn independently and uniformly at random from the response space. Unique in this case only restricts the requirement with regard to repetition of identical queries, which according to the reliability property give the same response. So for another challenge on the same instance, the same challenge on another instance, or a combination thereof, the response appears to be an arbitrary element in the response space. This supports unclonability with regard to imitation of the CRB, because it makes any model that aims to predict CRPs difficult, ideally prohibitively difficult. It also completes the analogy of PUFs and random oracles.

The property hinders imitation in three directions: Even if an attacker obtains a large number of devices that are built using the same PUF design, the attacker's advantage in guessing the responses for another device is negligible. In the same way it only gives negligible advantage to collect a large number of CRPs for a particular PUF instance if the response to another challenge that is not in the collection is demanded. Finally, in the case where the attacker knows parts of the response and is required to guess the remainder, the knowledge about the first part provides only negligible advantage. In mathematical notation with $|\mathcal{X}|$ being the cardinality of the response space, an ideal PUF satisfies

$$\mathrm{P}\left(\underline{x}_{c,d} = \underline{x}_{c',d'}\right) = \begin{cases} 1 & c' = c \wedge d' = d, \forall c, d \\ \frac{1}{|\mathcal{X}|} & \text{otherwise} \end{cases} \tag{1.10}$$

and there exists no simpler model of CRPs than a complete list.

An analogy that makes this property – together with reliability – particularly easy to grasp is to consider the PUF a keyed OWHF, where the manufacturing variations of the physical object pose the secret prefix that is fed into the otherwise deterministic OWHF before the challenge. Unpredictability among challenges and between parts of the response is then provided by the OWHF and the prefix that is different for each device, but constant after production, ensures unpredictability among devices at the same level.[4]

[4]In fact, this construction – extraction of a secret binary string that is used as secret prefix together

So unpredictability means it is sufficiently hard for an attacker to predict the response in any of the three directions described above, e.g. because the distribution of responses is sufficiently close to uniform and any dependencies among them are sufficiently weak so that a sufficiently accurate model comes close to an entire CRP list in complexity. Similar to the previous properties, metrics may rank PUF designs in their hardness of prediction and relate it to attacker models or security levels.

1.3. Typical Applications

1.3.1. Overview

The properties of PUFs enable several use cases. One of the first applications was authentication of physical objects by Pappu [2] with a resin slab that contains reflective particles. It was intended to be a cheaper alternative to holograms or digital identification ICs to protect against counterfeit products. PUFs as part of ICs started as a mere identification circuit to provide intrinsic identifiers [5], where it was sufficient to have unclonability against accidental replication. However, cryptoless authentication, in particular for devices with low computational power, soon became and remains a typical motivation in PUF literature [7, 13, 15], [16, 23]. A quite targeted example for cryptoless authentication is to integrate a PUF into a central-processing unit (CPU), where it can be used to prove code was executed on a specific piece of HW or to bind code to run only on the HW it has been licensed to [7]. Despite the academic interest, commercial availability remains limited: The startup Verayo Inc. once promoted a PUF based authentication tag, but it is no longer available. As a secondary use, the noise during evaluation of the PUF can serve as source of true randomness in a TRNG or true random seed for a pseudo-random number generator (PRNG) [29], [30]. Secure key storage was introduced in 2005 [16] after Lee et al. presented key generation in 2004 [12]. This use case soon gained great attention in the PUF field, cf. for example [17, 31]–[44]. Meanwhile, it is one of the most prominent use cases for PUFs in academia and the only use case commercially available from established HW vendors, such as Microsemi in their SmartFusion2 and IGLOO2 field programmable gate array (FPGA) families [45], Intel in their Stratix 10 FPGAs and system-on-chips (SoCs) [46], and Xilinx in their Zynq UltraScale+ devices [47]. Further use cases include an encryption algorithm secured against side-channel leakage by a PUF [48], and oblivious transfer based protocols from PUFs [49]. The two most typical use cases, cryptoless authentication and key storage, will now be presented in more detail.

with a cryptographically secure OWHF – can be a workaround for the lack of unpredictability among challenges found in many real-world electronic silicon PUFs, cf. Sec. 1.3.2.

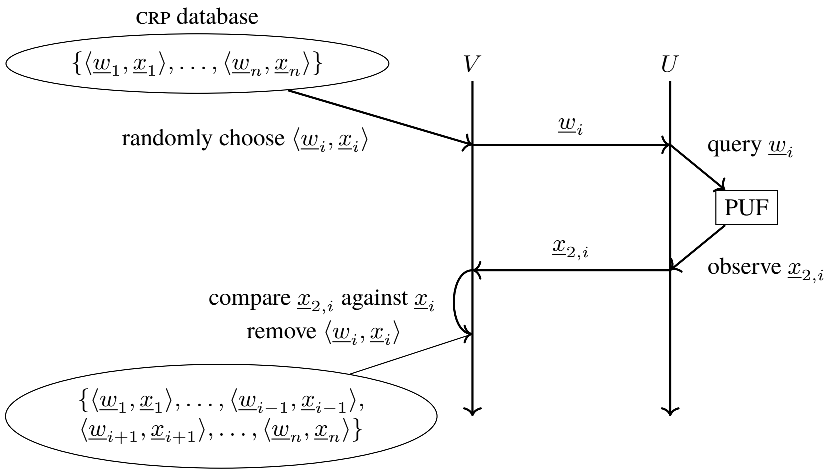

CRP database
$\{\langle \underline{w}_1, \underline{x}_1\rangle, \ldots, \langle \underline{w}_n, \underline{x}_n\rangle\}$
V
U
randomly choose $\langle \underline{w}_i, \underline{x}_i\rangle$
$\underline{w}_i$
query $\underline{w}_i$
PUF
$\underline{x}_{2,i}$
observe $\underline{x}_{2,i}$
compare $\underline{x}_{2,i}$ against $\underline{x}_i$
remove $\langle \underline{w}_i, \underline{x}_i\rangle$
$\{\langle \underline{w}_1, \underline{x}_1\rangle, \ldots, \langle \underline{w}_{i-1}, \underline{x}_{i-1}\rangle,$
$\langle \underline{w}_{i+1}, \underline{x}_{i+1}\rangle, \ldots, \langle \underline{w}_n, \underline{x}_n\rangle\}$

Figure 1.2.: Simple challenge-response protocol to authenticate a physical object using a PUF. For brevity, only the challenge indices are shown and index 2 denotes the response from an evaluation in the field rather than those in the CRP database.

1.3.2. Cryptoless Authentication

In this application, party V wants to verify that party U is in possession of a specific instance of PUF. This might be the case, for example, if the PUF is embedded into a credit card [16] where it can prevent card duplication, given the incapability to move the PUF to another card without damage and thus changed CRB. Several other scenarios can be thought of and are described in numerous papers. The protocol is based on canonical challenge-response authentication, where the security lies in the fact that only the authentic party is able to provide the correct response to the given challenge. Without a PUF, this can be implemented e.g. by a request to encrypt the challenge, which is a nonce, with a pre-shared secret key. In this case only a party that is in possession of the secret key is able to produce the correct response. The expected properties of a PUF allow for such a protocol without the use of encryption or other canonical cryptographic primitives:

At the beginning of the protocol, V has access to a database with a sufficient number of pre-recorded CRPs for the PUF to be identified, cf. Fig. 1.2. This database was created during a so-called *enrollment* phase in a confidential and trusted environment, such as the vendor's post-production testing site. V then selects one challenge from the database at random and sends it to U. U queries the PUF with it and sends the response back to V. V compares if the returned response is sufficiently similar – equality would be too strict due to the inherent noise in PUF responses – to the response in the database and removes the CRP unconditionally from the database. Depending on the result of the comparison, V either deems U authenticated, not authenticated, or decides to repeat the protocol to decrease the chance that U returned a sufficiently similar response by chance or a too unsimilar response due to a strong noise event.

The properties used in this scenario are: Reliability, because it first of all enables to recognize a pre-recorded CRP. Unpredictability then ensures that an attacker has only a negligible chance to guess the correct response without access to the respective instance of PUF, of which there exists only one due to unclonability. The property of being a function, together with a sufficiently large challenge space and unpredictability between CRPs, finally allows to perform the authentication more than once without a confidentiality requirement on the channel.

The promoted advantage that no canonical cryptographic primitives are required was considered a key solution to the upcoming internet of things (IoT), where strong security on heavily constrained devices is required. The constraint device, in above description U, merely needs to receive a message, query its PUF, and send a message, which is a task doable even with very few memory and computing power.

The application suffered a severe setback, though, as ML attacks revealed that most multi-challenge PUFs do not have sufficient unpredictability among CRPs, cf. Sec. 1.2.4. This allows an attacker to eavesdrop on the CRPs exchanged during an authentication in plain, and create a model that imitates the CRB without ever being in possession of the actual PUF equipped device. Furthermore, due to the limited size of the list of pre-recorded CRPs, an attacker could learn the challenges in the verifier's database by attempting to authenticate, recording the provided challenge, and aborting communication, because the CRP is only removed from the database after comparison. If the attacker then obtains the corresponding responses by pretending to be the verifier for the PUF under attack, she is able to provide the correct response for every challenge in the actual verifier's database, thus imitate the CRB.

Although advanced protocols that address both issues have been published, with two examples given in the following, commercial interest turned towards key storage instead. In the slender PUF protocol by Majzoobi et al. [50], the PUF reveals only a random subset of responses to the challenges in the query. The authentic verifier knows the responses to all queried challenges and can thus recognize if the returned responses are a subset. An eavesdropper or fake verifier, however, would not know to which challenges the returned responses belong, thus fail to obtain CRPs for a LUT or to train an ML model. A device that pretends to be the legitimate PUF would not be able to provide a sufficient number of correct responses, and would thus not be authenticated. The noise bifurcation protocol by Yu et al. [51] instead turns the ability to build a model of a PUF instance into a feature. Instead of a list of CRPs, the verifier learns a model of the PUF to be authenticated via a one-time interface that is locked before the device exits the trusted environment. During authentication, the PUF equipped device intentionally adds noise to the responses in a certain way that deteriorates the information provided to an eavesdropper while someone in possession of the PUF model such as the legitimate verifier is able to filter and analyze the noise, and thus to recognize if the noisy response was produced by the PUF in question or not.

1.3.3. Key Storage

If a device is not just supposed to be authenticated over a public channel, but able to establish a secure channel that provides confidentiality and integrity, canonical cryptographic primitives are necessary. The protection of the corresponding cryptographic keys on embedded devices against physical attacks is one of the most important issues in the field of embedded security. The requirement of low cost and the development of more sophisticated attacks led to several rounds of improvements from mask read-only memory (ROM) over eFuses and anti-fuses to embedded flash and complementary metal-oxide-semiconductor (CMOS) NVM, but all of them remain attackable one way or the other, even with the device powered-off [52]. The last point suggests the use of PUFs, given that their response is only present after the challenge has been fed through the PUF. For any electronic PUF, this requires the device – or at least the PUF – to be powered. Therefore the security of key storage can be raised to the problem of breaking the PUF's unpredictability or unclonability – or attacking a powered-on device, which falls into the scope of SCAs, FAs, etc., but not key storage.

For an ideal PUF, the response could be used directly as a cryptographic key, because the unpredictability property ensures that it contains full entropy and the reliability property says that the same key is provided every time it is retrieved from the PUF. Real PUFs, though, require post-processing such as sparse coding or helper data systems (HDSs) to store a cryptographic key with sufficient reliability. A common figure in this regard is a wrong key probability of less than 10^{-6} [17].

Fig. 1.3 shows two of the most popular HDSs in the PUF field, *fuzzy commitment* and *fuzzy extractor* [53]. Both operating largely the same way, the sole difference is whether an error-corrected version of the PUF response is restored for key derivation (fuzzy extractor) or a pre-defined key is restored (fuzzy commitment). Either case starts with an *enrollment* phase that is performed – as in the previously described application – in a confidential and trusted environment. With appropriate protections against e.g. side-channel and fault attacks in place and if the key is only used internally for storage encryption, this environment may be provided by the IC itself. There, a full entropy random binary vector $\underline{u}$ is encoded into a codeword $\underline{v}$

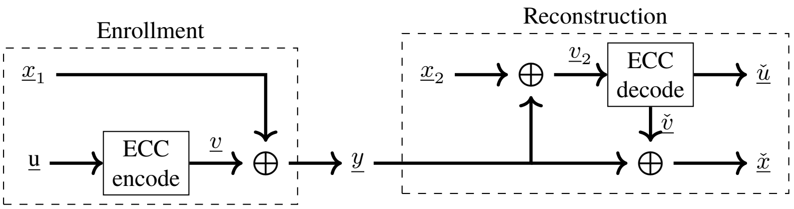


Figure 1.3.: Fuzzy extractor and fuzzy commitment for key storage with a PUF. $\underline{u}$ is a full entropy random binary vector. $\underline{x}_1$ is the PUF response during enrollment. $\underline{v}$ is a codeword of the ECC. $\underline{y}$ are the helper data. $\underline{x}_2$ and $\underline{v}_2$ are PUF response and codeword altered by noise. $\check{\underline{x}}$, $\check{\underline{v}}$, and $\check{\underline{u}}$ are error-corrected PUF response, codeword, and random vector, which are equal to $\underline{x}$, $\underline{v}$, and $\underline{u}$ with high probability.

of an ECC and combined in an exclusive-or (XOR) operation with a binary vector $\underline{x}_1$ produced by the PUF, to generate the helper data $\underline{y}$. If the cardinality of the response space is large enough, $\underline{x}_1$ is a single PUF response $\underline{x}_{a,c,d,e}$. Otherwise it can be an identifier comprised out of multiple responses. The ECC must be chosen strong enough to fully remove the distance between $\underline{x}_1$ and another binary string $\underline{x}_2$ produced in the field, except for the given key error probability. $\underline{x}_2$ is constructed the same way as $\underline{x}_1$, e.g. as $\underline{x}_{a',c,d,e'}$, but may still differ from it due to noise and a change in environmental conditions. Depending on the point of view, the XOR operation either encrypts the codeword $\underline{v}$ by $\underline{x}_1$ in a one-time pad style or vice versa. Though $\underline{u}$ is assumed to have full entropy, the dependencies in $\underline{v}$ introduced by the ECC make $\underline{y}$ provide some information about $\underline{x}_1$ to an attacker. If $\underline{x}_1$ has full entropy, the remaining entropy is that of $\underline{u}$. If $\underline{x}_1$ does not have full entropy, the remaining entropy is subject to how the dependencies in $\underline{v}$ overlap with bias and dependencies that make $\underline{x}_1$ not have full entropy. Given a sufficient amount of entropy in $\underline{x}_1$, though, the key remains strong enough even if $\underline{y}$ falls in the hands of an attacker. This allows to consider $\underline{y}$ public information and store it in some unprotected NVM before the device is shipped. During the *reconstruction* phase that happens in the field, $\underline{x}_2$ is XORed on $\underline{y}$ and the resulting noisy codeword $\underline{v}_2$ is fed to the ECC. In the fuzzy commitment scenario, it is sufficient that the ECC outputs the error-corrected message $\check{\underline{u}}$, which equals $\underline{u}$ as long as the noise was within specification. $\check{\underline{u}}$ can thus be directly used as cryptographic key if $\underline{x}_1$ had sufficient entropy. In the fuzzy extraction scenario, the ECC outputs instead $\check{\underline{v}}$, an error-corrected version of $\underline{v}$, which is XORed once again on $\underline{y}$ to obtain $\check{\underline{x}}$. $\check{\underline{x}}$ equals $\underline{x}$ as long as the noise was within specification, but given $\underline{y}$ it does not contain full entropy even if $\underline{x}_1$ did, so it must be fed into an entropy condensing function before it is suitable as cryptographic key.

1.4. Challengeability Classes

While PUFs can be classified by a manifold of characteristics, such as the physical phenomena used to measure the manufacturing variations, whether the evaluation instrument is integrated into the device that carries the PUF or external, or the ability to integrate it easily into ICs, the distinction between *weak* and *strong* PUFs is the most prominent one. Already the seminal thesis of Pappu [2] made such a distinction and since then many PUF related publications followed in this regard. However, the definition of what makes up a strong PUF has been changed more than once to include yet another new circuit design, since strong is subconsciously associated with *better* and circumvents the need to motivate why research about something that carries the attribute "weak" is worthwhile.

Pappu [2], who focused strongly on the authentication scenario, did not rely on the infeasibility to fully characterize the physical object. Instead, the physical processes that occur during evaluation should be so complex that a simulation becomes too hard to be worthwhile[5]. This complexity was deemed necessary to provide a one-way function, with unclonability and unpredictability as side-effects. Consequentially, the distinction between weak and strong PUFs was made on the complexity class of a simulation that calculates the response to a given challenge. Let $\underline{o} \in \{0,1\}^l$ be a suitable description of the physical object's manufacturing variations required by a simulator built for this type of PUF. Then a strong PUF according to [2] is one where calculation of the response $\underline{x}$ to some challenge $\underline{w}$ by simulation requires $\mathcal{O}(\tau^l)$ resources either in time or space, where τ is an arbitrary constant. The PUF can thus be efficiently scaled to counter the increase in computational power due to Moore's Law, because the simulation effort increases exponentially with the length of description of the physical object. A weak PUF is one where the simulation effort increases polynomially, i.e. a simulation as described above requires $\mathcal{O}(l^\tau)$ resources.

Tuyls et al. [16] used the terms instead to distinguish between a PUF for which an *upper* bound on the entropy of the response can be given (weak PUF), and those for which a *lower* bound can be proven (strong PUF). A further change in meaning was given by Guajardo et al. [17], who merely distinguished on the size of the challenge space. According to that work, any PUF that has "so many CRPs such that an attack [. . .] based on exhaustively measuring the CRPs only has a negligible probability of success" [17] is considered a strong PUF and a weak PUF is one where "the number of different CRPs N is rather small" [17]. Note that this classification is different from that by Pappu, as it only considers the cardinality of the challenge space, but neither takes into account the complexity of the physical object nor the

[5]This approach was later reinvented as so-called public PUFs, a form of simulation possible, but laborious (SIMPL) systems [54], [55].

complexity class of a simulation. The work instead assumes that any PUF provides unpredictability between CRPs and cannot be replaced by a simulation. Rührmair et al. [13] then combined the distinction based on the number of available challenges with a relaxation in required properties for weak PUFs. E.g. only a strong PUF requires unclonability in any of the previously described interpretations, including the infeasibility to built a complete CRP LUT in a reasonable period of time and the infeasibility to build a model from an obtainable subset of CRPs. Yet another definition for weak and strong PUFs is given by Armknecht et al. in [9], where, in a failed attempt to recount [17] correctly, the distinction is made on the number of CRPs required to build a sufficiently accurate model of the PUF. If this number increases exponentially in “some security parameter” [9], they call it a strong PUF, otherwise it is a weak PUF.

The notion of weak and strong PUFs has been criticized before, e.g. as confusing compared to the meaning of the terms in classical cryptography [9], and carrying the risk of pejorative or judgemental interpretation [24]. It is also confusing given that most strong PUFs are “broken” in the sense that they do not provide the expected unpredictability among CRPs, while weak PUFs remain strong enough to find their way into commercial products, cf. Sec. 1.3. To provide a more intuitive classification and avoid tendentious labels, this work uses the following terms:

Single-challenge PUF refers to those types of PUF that have one, potentially implicit, challenge.

Multi-challenge PUF refers to those types of PUF where the response depends on a challenge of fixed length and the set of valid challenges has cardinality larger than one.

Arbitrary-challenge PUF is a postulated class to capture the original idea of keyed physical one-way hash functions, where a challenge of arbitrary length determines an output of fixed size and there is sufficient *diffusion of entropy*, i.e. all parts of the response are affected by all parts of the challenge. As PUFs of this class are hash functions themselves, they could replace algorithmic hash functions in applications that benefit from the mapping being unique to the device. Note that realizations of this class presumably require one of the following to be possible: Either the physical object can be exposed to an arbitrary number of challenge signals at the same time, which must interact with each other and affect the output; or the challenge signals influence the state of the physical object in a reversible way, such that the influence lasts until the challenge is fully entered and the response recorded, but the initial state is reproducible to ensure that a challenge always leads to the same response no matter what challenges where used before. Future work may show whether PUFs of this class can be realized or not.

1.5. Examples of PUF Circuit Candidates

As a first example of a PUF, Pappu's optical PUF [2] has been introduced at the beginning of this work, because it is an intuitively understandable example and makes the concept easy to grasp. This section presents a selection of popular electronic silicon PUFs, since this category of PUFs has received the most attention both in research as well as commercially, in recent years.

1.5.1. RO PUF

In 2002, Gassend et al. founded the family of electronic silicon PUFs with the proposal of a "self-oscillating loop circuit" [7], reproduced in Fig. 1.5, which contained a configurable "non-monotonic delay circuit" [7], reproduced in Fig. 1.4. The proposed delay circuit is made up from a chain of identical stages, with as many stages as challenge bit positions minus one. The last challenge bit position selects which path is routed to the output of the delay circuit. Each stage consists of two parts that serve different purposes: The multiplexers provide configurability by connecting the upper and lower path either through or crossover, so the number of different paths through the delay circuit is exponential in the number of stages. Since each component in the delay circuit is affected by manufacturing variations that determine their exact propagation delay, the exact delay through a chain will vary by the path taken, so by challenge, and from chain to chain, thus also from device to device. The buffers in turn provide non-monotonicity, because the inner buffers are only enabled – increasing drive strength and thus reducing propagation delay – if the opposite input is currently low. Without the buffers, the delay of the entire chain would simply be the sum of delays through the multiplexers in the chosen connection and thus be easily modeled by an additive delay model. Using such a model, the individual delays of the multiplexers – which constitute the secret here – can be calculated from a small number of measurements of the entire chain delay, their number linear in the number of stages. So the PUF candidate could trivially be broken. The buffers are thus important for the unpredictability of this PUF candidate. Apart from that, optimization of propagation delay in ICs both in average value and in spread is a heavily researched topic, too, so it seems justified to assume that it is hard, even for a fab, to further reduce this variation. If the usual level of variation in propagation delay is measurable with sufficient precision to extract reliable information, it thus makes a promising PUF candidate circuit.

To measure small propagation delay variations, a self-oscillating loop circuit lends itself, because the resulting small variation in cycle duration can accumulate over time. So after observing the oscillation over a sufficiently long period of time and counting the cycles, an arbitrarily precise measurement of the cycle

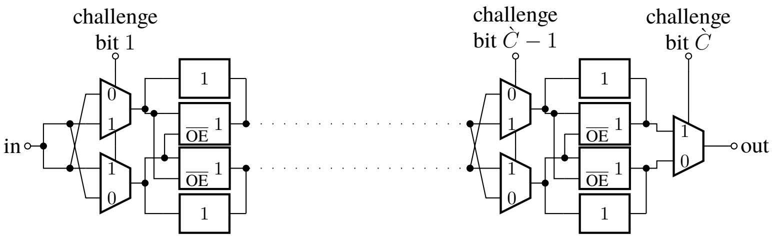


Figure 1.4.: Configurable non-monotonic delay circuit with $\grave{C}-1$ identical stages as proposed by Gassend et al. [7].

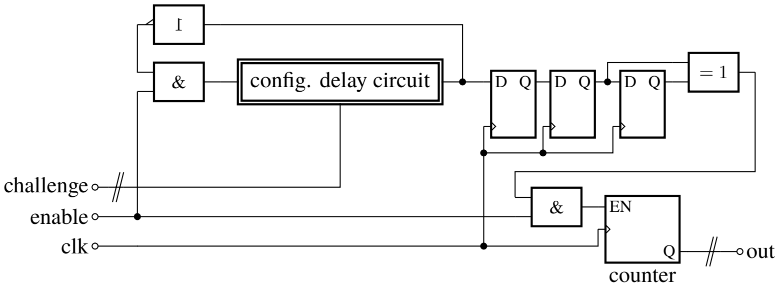


Figure 1.5.: Self-oscillating loop circuit as proposed by Gassend et al. [7].

duration can be made, theoretically. In practice, this only holds as long as the cycle duration remains constant or its change can be compensated for, such as temperature dependency that is compensated using an additional temperature sensor. Instead of a costly temperature measurement and correction calculations, and since there may be other environmental conditions that affect actual cycle duration, Gassend et al. proposed the well established solution of differential measurements. More precisely, they proposed to use the ratio of frequencies of two loop circuits placed next to each other, or of two unequal challenges applied to the same loop circuit consecutively, as a response. The approach of differential measurement is nowadays found in virtually every electronic silicon PUF.

Because delay measurement with such a self-oscillating circuit was rather slow, and the oscillation might be seen in a differential power analysis (DPA), Gassend et al. [23] proposed to use an arbiter instead, a circuit that precisely measures which of its two inputs rose to 1 earlier. With the rightmost multiplexer in Fig. 1.4 replaced by the arbiter and another delay stage inserted before it to process challenge bit $\grave{C}$, the output of the arbiter provides a differential measurement of the delays in both paths through the chain.[6] Although the measurement is rather coarse, since it only measures which path has the smaller delay, but not by how much, the benefits outweighed this. The concept earned a name of its own as the *arbiter PUF*, which is further described in Sec. 1.5.2. For several years, the arbiter PUF was the dominating electronic PUF, while the original RO PUF faded out of community memory.

In 2007, Suh et al. [32] reinvented the RO PUF without configurable delay circuit as a putatively separate approach from the arbiter PUF. The necessary response entropy was not achieved by application of a series of challenges to the same loop, but from a large array of fixed length inverter chains, whose frequencies are measured and later compared against each other, see Fig. 1.6. This design is what is nowadays typically considered an RO PUF. While the overall size of such an RO PUF can be minimized by multiplexing the outputs of the ROs onto a single cycle counter, the area per response bit is still worse than for the arbiter PUF and evaluation takes longer. However, this was deemed acceptable to overcome the claims of predictability that meanwhile arose regarding the arbiter PUF, since the response to different challenges is not independent due to the reuse of path elements and may be biased due to unequal routing of the delay circuit. Separate, identically routed and thus apparently unbiased ROs can be easily achieved in standard FPGA integrated development environments (IDEs) by using hard-macros or precisely constrained placement and routing of all parts of the ROs. Note that as this design became the typical RO PUF, several publications later claimed to

[6]Note that $c \in \{1, \ldots, C\}$ iterates challenges, while $\grave{c} \in \{1, \ldots, \grave{C}\}$ iterates bit positions in a challenge.

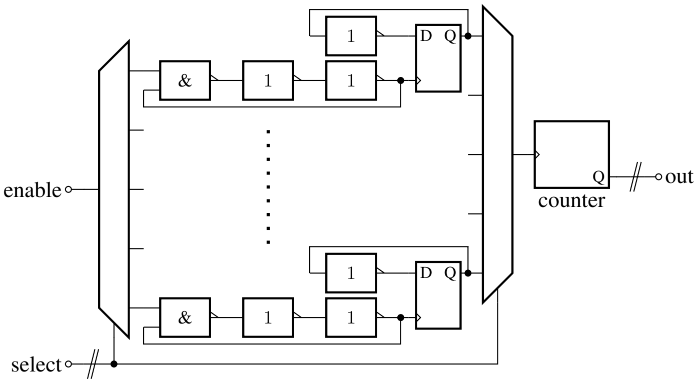


Figure 1.6.: An array of unchallengeable flipflop buffered ROs that are multiplexed to a single counter for frequency measurement as example of what is nowadays referred to as RO PUF. One of several designs used at TUEISEC. The number of inverters in the ring, multiplexers, and counters, and the type of buffering differ between authors.

have invented a challengeable RO PUF, e.g. [56]–[58], although this was already a property of the original design by Gassend et al., as Maes et al. mentioned in 2012 [34].

The RO PUF design of Suh et al. is sometimes considered a multi-challenge PUF, because a challenge may select which pair of ROs to compare, but following the definitions in this work, it is rather a single-challenge PUF comprised out of multiple cells that each produce a single response bit. The reason is twofold: First, the interaction between challenge and the manufacturing variation affected circuits is rather trivial. Second, while the number of *possible* challenges grows quadratically with size, i.e. with the number of ROs in the array, only few of them are *permissible* to keep response bits independent. As Suh et al. already noticed, even with perfectly identical and independent ROs, the entropy in all response bits based on which RO is faster than some other RO, is limited by the number of orderings that can be made from a given number of ROs [32]. So while $\frac{N(N-1)}{2}$ pairwise comparisons between N ROs can be made that each return one bit of information, some of them are redundant as they collectively describe which of the $N!$ possibilities to sort N ROs by frequency applies, an information that can be represented by $\log_2(N!) \leq \frac{N(N-1)}{2}$ bits, where equality only holds for $N \leq 2$.

In practice, the assumption of independent and identical ROs does not hold due to regional effects in manufacturing variations, which means that ROs in one area of the die collectively tend to have a different frequency than those in another area of the die. Such effects are called spatial artifacts, cf. Sec. 4.2. Together with a reduction in overall entropy, this makes the outcome of comparisons of spatially distant ROs particularly predictable, which led to the recommendation to place the ROs closely together in an array and solely compare adjacent ROs by Maiti et al. as early as 2009 [56]. However, dense placement can lead to dependencies that arise during readout, as Costea et al. have shown in [59]: The frequency of ROs can be affected by surrounding logic, even if it is not actively switching, but in particular if it does switch with approximately the same frequency. This may cause ROs to drift in frequency towards, and possibly fully lock-in with, an adjacent clock signal or another RO that is placed sufficiently close, which would reduce the measured frequency difference, thus impair reliability of readout. Furthermore, the conclusion in [56] that the resulting $N - 1$ response bits, obtained by overlapping pairwise comparison (RO$_1$ vs. RO$_2$, RO$_2$ vs. RO$_3$, RO$_3$ vs. RO$_4$, and so forth), are independent, does not hold either. Except for some pathological distributions of frequency, given e.g. that RO$_2$ has a higher frequency than RO$_1$ increases the chance that it will also have a higher frequency than RO$_3$, thus if the first comparison resulted in a 1 as response bit, the second bit can be guessed as 0 with chance better than $1/2$. This shows up as strong negative spatial correlation when the response bits are considered to be located at the place where the ROs compared to obtain this bit touch each other [**60**]. Thus pairwise comparisons need to be non-overlapping. If, instead of mere pairs of two ROs, moderately sized groups of ROs are compared together, their order in terms of frequency can be efficiently decoded into a multi-bit subresponse as shown by Yin and Qu [61]. As long as the ROs in a group are located densely together to reduce the effect of spatial artifacts mentioned above, this method can extract more entropy than non-overlapping pairwise comparisons and can do so efficiently, because it can directly map each of the $M!$ possible orders in a group of M ROs onto a $\lceil \log_2(M!) \rceil$ bit subresponse. However, the produced subresponse will contain biased bit positions unless $M!$ is a power of two.

In the presence of SCAs, the oscillating nature of RO PUFs becomes a curse, because the exact frequency of all currently running ROs is easily observed in the emanated electromagnetic (EM) spectrum or power consumption, which was one of the reasons for Gassend et al. to favor their arbiter PUF over their RO PUF. Thus the absolute or relative frequency looses all secret information, and the only remaining secret is the attribution which peak in the spectrum belongs to which RO. At this point, the overlapping pairwise comparison is most vulnerable, because as Merli et al. showed in 2011 [18], it is trivial to attribute the peaks in the spectrum to an RO even without localized measurements, because for consecutive comparisons where only the currently compared ROs are activated, one of both peaks always remains, which thus must belong to the RO that is shared among these two comparisons. This reduces the entropy of the entire response to at most a single bit. For non-overlapping pairwise and group order based approaches, security depends on the number and placement of cycle counters. With just one counter, the order in which the ROs are measured has to be randomized, otherwise the security is reduced to knowledge of this order. Solutions with multiple counters can be attacked by localized EM SCAs, because the cycle counters are found to be the strongest source of EM emanation, but allow the countermeasure to randomize the counter an RO is measured with instead of the measurement order [19].

1.5.2. Arbiter PUF

The arbiter PUF was developed by Gassend et al. [23] as an improvement over the original RO PUF design that is less prone to SCAs and faster to evaluate. It uses a configurable delay circuit similar to the original RO PUF, though not as a single configurable delay element to determine the frequency of a self-oscillating circuit, but as two conjointly configurable parallel delay paths, where the response reflects which of the two paths has the smaller propagation delay. The necessary modifications are small. Compared to Fig. 1.4, it is sufficient to replace the final multiplexer with a regular delay stage and add an arbiter circuit at the end that is able to report which of its two inputs rose to a high level earlier. The number of paths through the circuit is still exponential in the length of the challenge and the paths are still comprised of a small set of path elements whose size grows only linear with the challenge length. To measure the propagation delay in the paths, a rising edge is applied to the enable input, which is the starting point of both paths. The edge then races against itself on both paths until it reaches the inputs of the arbiter circuit. Since the arbiter reports which of its inputs rose earlier, it effectively reports which of the two paths had the smaller propagation delay. This provides the additional benefit that it inherently performs a differential measurement and does not require explicit comparison of individual measurements such as with the RO PUF according to Suh et al. [32]. The basic arbiter scheme as published in [23] is reproduced in Fig. 1.7.

To design a good arbiter is difficult because it has to correctly report without metastability which input rose earlier, even if both inputs rise only some ten picoseconds after another. It should also be free of skew as this would introduce bias into the response bits. In the original design by Gassend et al. [23], a transparent latch with low-active enable input has been used, cf. Fig. 1.7. Since this causes a skew due to the setup time of the latch, they fixed some of the challenge inputs to lengthen one of the paths to compensate the skew. Instead of a latch, a regular D-type flip-flop may be used, though with the same issue of skew due to the necessary setup time. Skew may also arise from unequal routing of the traces that lead to the arbiter, in particular on FPGAs, where only a predefined set of routing resources can be used that is not intended to serve a flip-flop’s clock

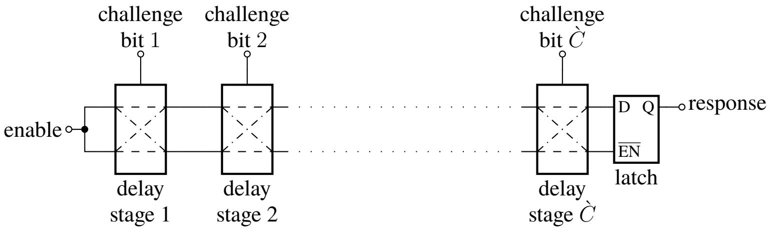


Figure 1.7.: Basic arbiter PUF circuit with $\grave{C}$ identical differential delay stages and a final arbiter as proposed by Gassend et al. [23].

input from common logic signals rather than the highly optimized clock tree.

In [7], Gassend et al. emphasized multiple times the importance of the tri-state buffers, cf. Fig. 1.4, to make their configurable delay circuit non-monotonic and render additive delay models inapplicable. However, the experiments in [7] are performed with regular buffers, presumably because tri-state buffers are hard to realize on an FPGA, and the arbiter PUF is proposed in [23] with regular buffers, too. This would be a plausible explanation why the simple modeling attack also provided in [23] was able to predict a device's response with a failure rate only slightly above those from the PUF's own measurement noise. The tri-state buffers are instead mentioned as one of muliple ways to harden the arbiter PUF against model building attacks with additive delay models, among those also the better known feed-forward arbiter extension, cf. Fig. 1.8. While the decision to remove tri-state buffers had benefits such as FPGA based implementations, and the effectiveness of tri-state buffers to harden arbiter PUFs remains unproven, it coined the term arbiter PUF to the simple mux-and-buffer design, which is prone to model building attacks. Arbiter PUFs have therefore expectably been the target of manifold publications that improved this kind of attack either by a lower error rate or fewer CRPs required for calculating the model's parameters respectively training an ML approach, see for example [13, 62], [63].

Several modifications and extensions of the basic arbiter PUF have been published since. The previously mentioned feed-forward arbiter PUF, reproduced in Fig. 1.8, contains additional delay stages, where the path selection input is not determined by the challenge, but by the output of an additional arbiter that has its inputs connected to an intermediate point in the chain. The number of feed-forward stages is limited though, since there need to be enough challenge controlled delay stages between the sample point of the additional arbiter and the controlled delay stage to allow the arbiter and delay stage reach its final state before the racing edges reach the arbiter controlled delay stage. The feed-forward extension has been broken together with the XOR-Arbiter extension and some other PUF candidates

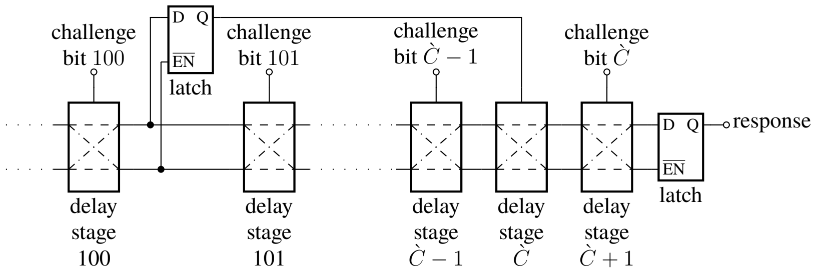


Figure 1.8.: Feed-forward arbiter PUF circuit with one feed-forward stage as proposed by Gassend et al. [23].

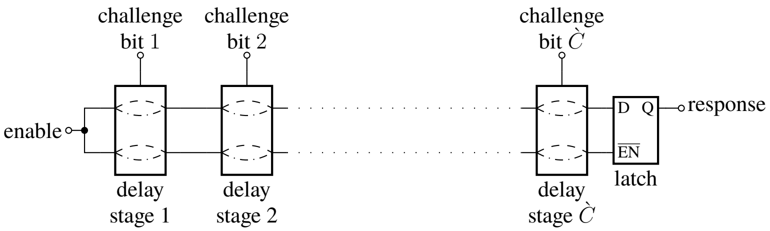


Figure 1.9.: Non-crossing arbiter PUF circuit as proposed by Majzoobi et al. [64].

in e.g. [13]. Majzoobi et al. [64] proposed a modification that aims to further ease implementation on FPGAs, cf. Fig. 1.9. They noticed that it is difficult on FPGAs to balance the routing within the delay stages due to the cross-over of paths if the corresponding challenge bit is set, since this requires a different type of routing resource to be used. They thus proposed a non-crossing design that can use routing resources of equal type for those path elements that require balancing while still matching the simple delay model from [23].

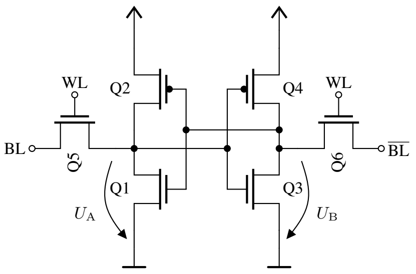
A notable recent extension of the arbiter PUF is the so-called interpose PUF, which uses the output of an XOR PUF to determine an additional challenge bit to another XOR PUF, whose remaining challenge bits are identical to those of the first XOR PUF. This design is claimed to be secure against all known model building and ML attacks to date [65].

1.5.3. SRAM PUF

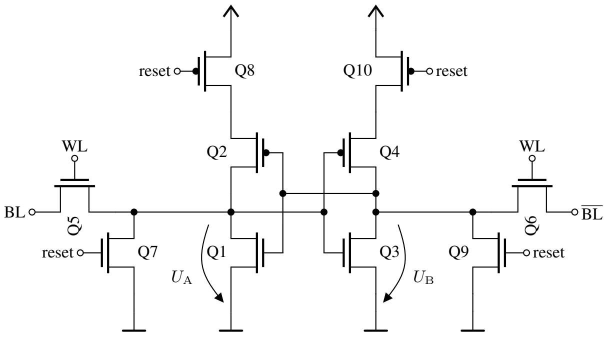
In contrast to the previous examples of delay-based PUFs, which utilize the manufacturing variation in propagation delay, the SRAM PUF concept utilizes the manufacturing variations in *threshold voltage*. Since threshold voltage is also a crucial parameter for the performance of ICs, similar amounts of research and development are devoted into precisely controlling it during production. Still, threshold voltage variation is expected to rise for nanometer scale processes, because transistor volume becomes so small that only a few dozen doping atoms fit in it, thus each doping atom more or less already causes a correspondingly larger change in threshold voltage. Threshold voltage therefore appears to be a good basis to utilize in a PUF circuit, just as propagation delay does.

The use of threshold voltage variation originates from Lofstrom et al. [5] in 2000, who evaluated the drain currents of an array of transistors under equal gate-source voltage against their mean using an autozeroing comparator. To avoid the bulky analog part of this mixed-signal design and to improve power consumption and read-out speed, Su et al. [6] used cross-coupled NOR gates instead, see Fig. 1.10b, which provide positive feedback to each other and are thus able to amplify and evaluate their own threshold voltage differences. Su et al. introduced it as cross-coupled NOR gates, but already recognized that their design "is similar to a 128b SRAM array" [6]. Shortly after this was published in February 2007, Holcomb et al. [66] (in July) and Guajardo et al. [17] (in September) both found out that SRAM in commercial of-the-shelf (COTS) microcontroller units (MCUS), FPGAS, etc. provides almost the same quality of response without the need for an ASIC design. Although COTS SRAM has the drawback – compared to the design by Su et al., which can produce its response at any time – that the response is only available after power-on until the SRAM is initialized, it became one of the most widely used types of PUF nowadays.

Compared to the cross-coupled NOR gates by Su et al., a COTS six transistor SRAM cell consists of a pair of cross-coupled *inverters*, see Fig. 1.10a, but in both circuits, positive feedback works as follows: With transistors Q1 through Q4 perfectly matched, the DC operating point for both circuits is $U_\mathrm{A} = U_\mathrm{B} = \frac{U_\mathrm{DD}}{2}$, so all four transistors operate in saturation region and cause excessive currents in both of their branches. This state, though, is an unstable equilibrium. A small decrease of U_A lowers the drain current of Q3 and increases that of Q4. The resulting current imbalance increases U_B, which brings the drain currents of Q1 and Q2 out of balance, too, but in the opposite direction, so U_A decreases. Since this was the starting point, it closes the positive feedback loop and eventually, $U_\mathrm{A} = 0\,\mathrm{V}$ and $U_\mathrm{B} = U_\mathrm{DD}$, in which case Q1 and Q4 are in cutoff region and Q2 and Q3 are in ohmic region. The same mechanism applies the other way round if U_A is slightly increased rather than decreased from equilibrium. Once either stable endpoint is reached, it requires a strong external current to force node A or B away from GND or VDD and above or below, respectively, half the supply voltage, in which case the positive feedback takes over and flips the SRAM cell to the other stable endpoint. During regular operation as a memory, this current is delivered by the write amplifier via BL, Q5, $\overline{\mathrm{BL}}$, and Q6.



(a) COTS SRAM PUF cell.



(b) PUF cell according to Su et al. [6].

Figure 1.10.: Simplified schematics of two types of SRAM PUF cells.

For the use as PUF, what matters is the power-up behavior of the circuit, which is explained and depicted in e.g. [67]: As supply voltage rises in Fig. 1.10a, respectively the reset signal is lowered in Fig. 1.10b, Q2 and Q4 turn on and their drain currents increase U_A and U_B until the drain currents of Q1 and Q3 compensate those of Q2 and Q4. Due to manufacturing variations, U_A and U_B will not be equal at this point, which makes the cell flip into one of its stable endpoints due to the positive feedback mechanism described above. The period of time that it takes for a cell to decide for either endpoint depends on the amount of mismatch between the transistors, where more mismatch makes the cell decide faster. If there is only few mismatch between the transistors, the cell may remain in the unstable equilibrium long enough for noise to decide which endpoint is reached.

To avoid the latter and increase reliability, Bucci and Luzzi [68] suggest a circuit where amplification of drain current mismatch can be performed without positive feedback and only once U_A and U_B have diverted far enough, evaluation of their difference into a logic value is triggered by another external signal.

In contrast to RO and arbiter PUFS, only few attacks have been published on SRAM PUFS. One of the most notable attacks by Helfmeier et al. observes the photons emitted by conducting MOSFETS [28].

1.6. Evaluation of PUF Candidates

To fully evaluate whether a PUF candidate circuit fulfills the properties introduced in Sec. 1.2, multiple levels of assessment are required. As the scope of this work will be limited on metrics to evaluate the CRB of a PUF candidate circuit, this section briefly outlines other levels of assessment as context.

Starting from the bottom up, a close inspection of the overall circuit design similar to the requirements of TRNG certification is necessary. This can validate whether the output is indeed the result of an interaction of the challenge with the physical object and not dominated by a parasitic effect of the evaluation method. It also protects against fake PUFS that are sold as black box design and produce their CRB not based on queries to a physical object, but contain some block cipher with device specific keys known to the vendor or leakable through a backdoor. Analysis from this point of view is mostly based on physics and knowledge about manufacturing processes and their typical variations. Notable examples can be found in the works of Pappu [2], Tuyls et al. [16], and Skoric et al. [21].

The next aspect then is to analyze on a similar physical level whether the property of unclonability holds, and on which level. This requires research about alternative measurement and manufacturing methods and experiments in replication of the physical object. Depending on the required level of unclonability, it may also include tests on whether it is possible to tweak a PUF instance to be a replica of another one, as well as tests to build an imitation of some instance, e.g. by replacement of the PUF through a HW implementation of a model of another instance's CRB. Since the number of possibilities to explore for this aspect can be very large, analysis will be limited to the most promising ways of attack in practice, though.

A third topic of assessment is the security against physical attacks such as SCAs and FAs. Here, the results of the inspection of underlying physical processes can provide a head start e.g. through identification of worthwhile points of measurement or attack. The target of such attacks differs by use case. For the key storage scenario, observation of the response through a side channel as in e.g. [20, 28] already marks a successful attack. For the authentication use case, where CRPs are exchanged in public, a FA or SCA should not provide additional information that could help to build an imitation of the PUF instance.

Finally, the statistical properties of the CRB have to be evaluated. They determine how well a PUF candidate fulfills the properties of reliability and unpredictability, which also affects the ability to produce imitations. Analysis here includes bias and correlations among the responses of multiple devices that would enable to predict the response of a device more easily. It also covers the probability of error within the response, which is important e.g. to select appropriate error correction for the key storage use case. Within this area of research, the issue of predictability along the dimension of challenges emerged as a partially independent subfield. Instead of statistical methods, it focuses on ML to build a model of the CRB of a multi-challenge PUF, which then can serve as imitation. It is typically trained from previously recorded CRPs of the same device and shall then predict the response to yet unseen challenges with high probability of being correct. A frequent issue at this point is that a small part of the challenge bit positions has major influence on the response, which led to the work by Ganji et al. [69] about so-called k-junta functions.

1.7. Contribution and Structure of This Work

As mentioned in the previous section, the focus of this work lies on the statistical evaluation of reliability and in particular unpredictability. Although some of the metrics discussed herein can help with ML attacks on multi-challenge PUFs, the scope of this work is on statistics, not ML.

As a starting point for the remainder of the work, Chpt. 2 provides a comprehensive overview of existing metrics for the reliability and unpredictability of PUFs. It is followed by a thorough review and evaluation in Chpt. 3, which reveals misunderstandings with regard to sensitivity to various types of flaw, overlap between metrics, and caveats when using compression or TRNG test suites for PUFs without special care. It also introduces the statistical models that are the foundation for confidence intervals and hypothesis tests in the remaining parts of the thesis. The use of multivariate models allows to resemble permissible flaws of real-world datasets without invalidating the tests. As evident from Chpt. 2, most state-of-the-art metrics are either not explicitly based on any statistical model, which limits their informative value, or on a mere univariate Bernoulli source, which regularly oversimplifies the CRB of real-world PUF candidates and causes incorrect security claims as demonstrated in Chpt. 3, too. Chpt. 4 finally takes a novel approach to security evaluation of PUF candidates through statistical hypothesis tests, where it introduces an initial set of tests including tests for spatial autocorrelation and the extension of selected common metrics into hypothesis test. It also extends the expected conditional min-entropy as proposed by Delvaux et al. [36] to multivariate statistical models, which strongly broadens its applicability.

1.8. Notation

1.8.1. PUF Data and Dimensionality

As introduced before, binary response data of PUFs is denoted by the small Latin letter x in this work. Where necessary, e.g. for metrics to measure the performance of bit extraction algorithms, the small Greek letter ξ denotes the (digitally sampled) analog response values produced by a PUF candidate circuit. The so-called true response of either binary or analog form is indicated by an overset bar: $\bar{x}$, $\bar{\xi}$

Both symbols typically show up either with a series of indices or as multidimensional array, though, because PUF experiments produce highly dimensional data. In contrast to e.g. TRNG tests, where the result typically is a small number of one-dimensional vectors acquired from multiple power-ups of a few test devices, the test of a PUF candidate may result in a dataset of up to five dimensions. First, the response of a single access to a single test device may produce a multi-bit response. Because PUFs are inherently noisy, multiple accesses to each device have to be made to investigate the amount of noise this PUF candidate suffers from, makes two dimensions yet. A PUF typically digests a challenge that influences its response, which gives a third dimension. The CRB of a PUF has to be unique for a device, which requires to test and compare the responses of a sufficiently large number of test devices, makes four dimensions. A fifth dimension arises if a PUF candidate is tested over varying environmental conditions. To support such a high number of dimensions in an intuitive notation, the number of dimensions is denoted by as many underlines below the symbol. A five dimensional binary response dataset would thus be written as $\underline{\underline{\underline{\underline{\underline{x}}}}}$.

To operate on highly dimensional data, named indices are advisable. If the number of dimensions additionally may change, e.g. by taking the mean along one of them, they even become a necessity to avoid ambiguous formulae. The index symbols are the lower case version of the first letter of the names of the five dimensions, which are chosen to be easy to memorize:

1. Accesses, repeated evaluations for otherwise identical conditions to capture the noise during evaluation.
2. Bit positions, because a single response may be comprised of multiple bits, i.e. a binary vector itself.
3. Challenges, to capture how the response relates to various challenges.
4. Devices, to analyze how manufacturing variations change CRB between devices.
5. Environment, which reflects e.g. aging and operating conditions, because they can affect the statistical properties.

Thus lower case letters a, b, c, d, e provide an easy to grasp set of indices for all five dimensions, and capital letters A, B, C, D, E represent the respective number of elements in each of the five dimensions, typically indexed from unity onward, so e.g. $b \in \{1, \dots, B\}$. The actual number of dimensions present in the data depends on the type of PUF and purpose of evaluation, of course. For a typical arbiter PUF, which provides a single bit of output, bit positions is correspondingly a singleton dimension, so $B = 1$. The standard SRAM PUF is single-challenge, so the third dimension is singleton, i.e. $C = 1$. A preliminary test of a new design may only be tested in lab conditions to more quickly identify basic flaws, in which case the last dimension is singleton. Indices of singleton dimensions may be omitted for brevity, as long as it does not lead to ambiguity. Indices may also be omitted if the original definition of a metric did not consider the corresponding dimension and there is more than one reasonable way to extend it. If multiple indices for the same dimension are required, such as when devices are compared against each other, one or more primes are added to the respective index and end value, e.g. $d' \in \{d + 1, \dots, D' = D\}$.

To support metrics that operate on identifiers, which are a concatenation of responses to multiple challenges, additional indices k and l are introduced. They represent the identifier and the bit position within an identifier, respectively. Since the identifiers are concatenations of responses, each bit in an identifier belongs to some bit of a response to a particular challenge, and there is a bijective mapping between a (k, l)-tuple and a (b, c)-tuple. This allows a neat extension of the notation introduced above by replacing b and c by k and l. The capital letters again denote the number of elements, i.e. K is the number of created identifiers per device and each identifier contains L bit positions.

While $c \in \{1, \ldots, C\}$ indexes a challenge word, it is sometimes necessary to index the bit position within a challenge. This is done by $\grave{c} \in \{1, \ldots, \grave{C}\}$, i.e. a challenge word contains $\grave{C}$ bit positions.

Despite the high dimensionality of PUF data, a particular metric often operates on just one or two of the dimensions at once. Notation may then be simplified by matrix or vector operations, i.e. some dimensions are not indexed, but used to construct the vector or matrix. So a matrix that contains the entire response for all devices from a fully specified dataset would be written as $\underline{\underline{x}}_{a,c,e}$, whereas the corresponding response vector goes by $\underline{x}_{a,c,d,e}$. The number of underlines and the number of indices add up to the number of dimensions of the data. Vectors are row vectors if not explicitly stated otherwise. For matrices, the index that comes first in the alphabet determines the rows, and the second the columns, so the example above would have B rows and D columns. Transposition of a vector or matrix is denoted as $\underline{\underline{x}}^\top$.

1.8.2. Metrics

Metrics are represented by lower or upper case letters with various accents, where the letter typically indicates the general method used by the metric and the accents differentiate between different metrics that result form different ways to apply the method. For metrics based on a fractional Hamming-distance (FHD) where strings go along the dimension of bit positions, h is used, while other types of FHD, such as those with strings along the dimension of challenges or those that operate on identifiers, the letter is g. Entropy based metrics go by H. The fractional Hamming-weight (FHW) is also frequently used in metrics and due to its equivalence to the empiric mean for binary data, corresponding metrics use m as base letter. p denotes probabilities and q log-probabilities. Metrics related to correlations are based on r.

1.8.3. Sets and Spaces

Sets and spaces are written as calligraphic letters. For example, the response space of a PUF is $\mathcal{X} \subset \{0,1\}^*$ and the challenge space goes by $\mathcal{W} \subset \{0,1\}^*$. The cardinality of set $\mathcal{X}$ is denoted $|\mathcal{X}|$.

1.8.4. Statistics

Information theoretic sources can, for the scope of this work, be considered RVs. Such RVs are denoted by capital letters. For clarity of notation, realizations of the

RV use the corresponding lower case letter and the set of outcomes of the RV uses the corresponding calligraphic letter. For example, x is a realization of RV X, and $\mathcal{X}$ is the set of outcomes of X. The relationship between RV and realization may be made explicit where appropriate by $x \leftarrow X$.

Standard probability distributions are represented by capital Gothic print. The binomial and Bernoulli distributions with success parameter p are denoted as $\mathfrak{B}(N, p)$, with $N = 1$ for the Bernoulli distribution. The normal distribution, denoted by $\mathfrak{N}(\mu, \sigma^2)$, the beta distribution, denoted by $\mathfrak{Beta}(\alpha, \beta)$, and the Student's t distribution, denoted by $\mathfrak{T}(\nu)$, are used from Chpt. 3 onwards. The corresponding quantile distributions feature a superscript minus one, so the normal quantile distribution is $\mathfrak{N}^{-1}(z, \mu, \sigma^2)$, for example. The categorical distribution is denoted by $\mathfrak{C}(\mathcal{T} \mapsto \underline{p})$, where $\mathcal{T}$ is the set of outcomes whose elements have associated probabilities in the elements of probability vector $\underline{p}$.

To represent nonstandard distributions, f_T denotes the probability mass function (PMF) respectively probability density function (PDF) of some RV T, optionally dependent on some parameter vector $\underline{\theta}$. The corresponding cumulative distribution function (CDF) is denoted as F_T.

The Shannon entropy of a RV is calculated by $\mathrm{H}(\cdot)$ and the min-entropy by $\mathrm{H}_\infty(\cdot)$. $\mathrm{I}(\cdot, \cdot)$ calculates the mutual information.

1.9. Employed Real-World Datasets

Metrics for the reliability and unpredictability of PUF candidates can be compared on a symbolic level, e.g. for overlap between individual metrics, or the expectation of a metric for an ideal PUF. However, comparing their results for a given dataset can provide additional insight and increase confidence in the findings. For this purpose, synthetic datasets with specific defects are used throughout this work. They are produced by deterministic PRNGs from a known seed so that results are reproducible in line with good scientific practice. The corresponding Python code is available online and most easily found through an online search for the unique permanent ID of this work: db0e67612cbaf33b687df4c15a198025aead3392

While synthetic datasets are well suited to test the sensitivity of metrics to some known flaw, because they intentionally contain exactly this flaw and no other flaws, this approach requires knowledge about which flaws may exist and need to be tested. To research how metrics behave in the presence of multiple flaws and possibly identify new types of flaws that may exist in an actual PUF candidate, a selection of real-world datasets from silicon complements the analyses of this work. These datasets are introduced in the remainder of this section to avoid lengthy excursions later on.

1.9.1. SRAMsu

Dataset Description

SRAMsu is the dataset published by Su et al. in [67]. It is created from 19 test devices with an 8 by 16 (=128) array of custom SRAM-like identification cells in a 130 nm process ASIC. The schematic of the identification cells can be found in Fig. 1.10b. Two physical cell layouts are compared, a symmetric and a common centroid approach, both of them being well established methods from analog integrated circuit design to achieve very closely matched branches and thus minimal offset for operational amplifiers, comparators, etc. In this case, the design methods are supposed to best match the two branches of the identification cell, targeted at avoiding any bias towards a certain response value. Additionally, dummy cells of the same type are placed at the perimeter of the array to avoid edge effects.

Although the test chips contained a special supply network that allowed measurement of the analog offset voltage of the branches of the identification cells, this data is not publicly available. Instead, a table with the true binary response of each device as a hexadecimal string is published in [67]. No indication is given how true responses are defined, though.

Known Defects

As described in [67], the ratio of 1 responses per column increases from left to right for the symmetric design, meaning the leftmost columns tends to produce less 1 responses than the rightmost column. The common centroid design does not suffer from this behavior, but this comes at the price of doubled die space.

1.9.2. ROmaiti

Dataset Description

ROmaiti provides data for an RO PUF on a comparatively large number of FPGAs. It was made available for download by Maiti et al. as one of two datasets used in [70], which describes how the dataset was measured and provides an analysis of the dataset using metrics introduced in the same publication. The other dataset analyzed in [70], which contains only five devices, but includes temperature and supply voltage variation, is not used in this work. A comprehensive description of the dataset is given in the following paragraph based on [70].

ROmaiti contains 100 accesses to an array of 16 by 32 (=512) ROs, implemented in the midst of the fabric on Spartan3E S500 FPGAs (XC3S500E). Each RO was an instance of a hard macro to ensure equal routing and the five inverting elements

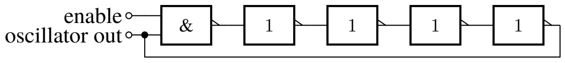


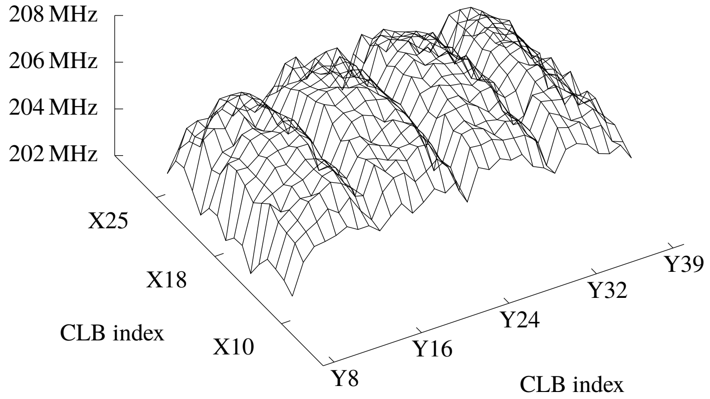
Figure 1.11.: RO design used by Maiti et al. [70].

of the ring were placed within a single configurable logic block (CLB) to ensure local routing. One of the five inverting elements was not an inverter, but a NAND gate to provide an enable input to the RO, cf. Fig. 1.11. The enable inputs and oscillator outputs, which are the two inputs of the NAND gate, were connected to (de-)multiplexers, so that for measurement, one RO at a time could be enabled and connected to a central 32 bit counter. While the enable signal of any RO was active, another 32 bit counter simultaneously counted the cycles of an on-chip 50 MHz crystal oscillator to serve as a reference frequency. Control of enable signals and readout of both counters was performed by a TCL script ran on a laptop computer via the joint test action group (JTAG) debug connection of the Spartan3 Starter Board that housed the FPGA. To obtain a large number of devices, Maiti et al. asked students of their university to provide the board they had to buy for course work of their computer engineering major, which resulted in 125 devices being measured and analyzed in [70]. Measurement was done in their computer engineering lab room, so presumably without explicitly controlled environmental conditions such as temperature or supply voltage. The overall measurement procedure for all 100 accesses to a device took less than two minutes. As of 2014, the dataset has been extended to 193 devices.

To produce binary response bits, different comparison strategies can be found in the literature for this dataset. In all these strategies, $\xi_{a,b,d}$ is the inferred frequency of the response signal observed in access a from the RO at position b on device d. So b serves here to index both a bit position when used with x as well as the position of a RO on the device when used with ξ. Maiti et al. [70] used an overlapping pairwise comparison, which means

$$x_{a,b,d} = \begin{cases} 1 & \xi_{a,b,d} < \xi_{a,b+1,d} \\ 0 & \text{otherwise,} \end{cases} \qquad \begin{aligned} a &\in \{1,\dots,100\}, \\ b &\in \{1,\dots,511\}, \\ d &\in \{1,\dots,193\}. \end{aligned} \tag{1.11}$$

Note that the direction of the comparison operator is a matter of choice, i.e. instead of $\xi_{a,b,d} < \xi_{a,b+1,d}$ one could also use $\xi_{a,b,d} > \xi_{a,b+1,d}$. As the dataset has been expanded since [70] was published, the direction used by Maiti et al. is hard to recognize. The overlapping comparison strategy leads to widespread negative correlation between response bits as explained in Sec. 1.5.1 and biased response bit positions in particular where comparison is performed across line breaks, i.e. if the rightmost RO in a row is compared to the leftmost RO in the next row of the



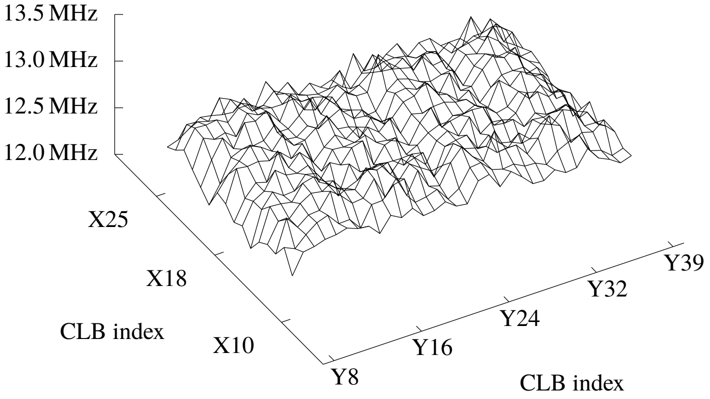


Figure 1.12.: Location preserving plot of mean (top) and standard deviation (bottom) over devices of true RO frequencies in dataset ROmaiti. Reproduced from [**60**].

array. To avoid biased response bit positions, Pehl et al. [**37**] omitted the response bits created by comparisons across line breaks, which yields

$$x_{a,b,d} = \begin{cases} 1 & \xi_{a,b',d} < \xi_{a,b'+1,d} \\ 0 & \text{otherwise,} \end{cases} \quad \begin{gathered} a \in \{1,\dots,100\}, \\ (b,b') \in \{(1,1),\dots,(15,15), \\ (16,17),\dots,(31,32), \\ (32,34)\dots,(480,511)\}, \\ d \in \{1,\dots,193\}. \end{gathered} \tag{1.12}$$

To furthermore avoid correlation among the response bits, a mutually exclusive pairwise comparison (MEPWC) strategy,

$$x_{a,b,d} = \begin{cases} 1 & \xi_{a,2b-1,d} < \xi_{a,2b,d} \\ 0 & \text{otherwise,} \end{cases} \quad \begin{gathered} a \in \{1,\dots,100\}, \\ b \in \{1,\dots,256\}, \\ d \in \{1,\dots,193\}, \end{gathered} \tag{1.13}$$

has to be used, as done in [**71**]. True responses are defined by Maiti et al. by taking the mean along accesses, then applying the comparison method.

Known Defects

Defects have not been described in [70], but the following have been identified during preparation of this work:

- Location dependent RO frequency that is affected by surrounding logic, see Fig. 1.12, which leads to bias in response bits.
- Spatial autocorrelation of RO frequency after taking the mean among accesses, presumably caused by speed gradients of varying direction and slope over the FPGA die, cf. Sec. 4.2.
- The frequency of all ROs changes gradually among accesses, which suggests intra-device distribution of RO frequency is affected by uncontrolled environmental conditions such as die temperature.
- Intra-device distribution artifacts for two devices, identified as a sudden strong change in RO frequency from one access to the next, presumably the result of a hang-up of data acquisition while environmental conditions continued to change.

The last two defects are visualized in Fig. 1.13 based on the deviation from true response

$$\xi'_{a,b,d} = \xi_{a,b,d} - \frac{1}{A}\sum_{a=1}^{A} \xi_{a,b,d}. \tag{1.14}$$

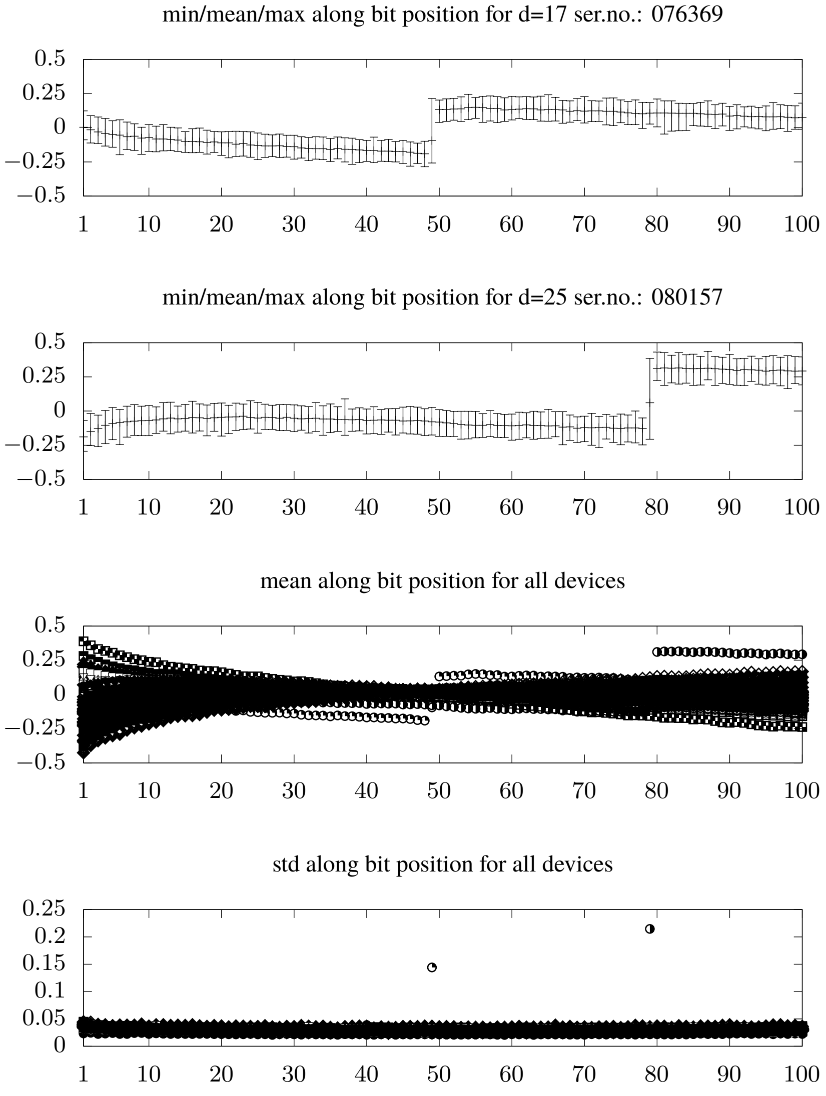


Figure 1.13.: Measurement artifacts in dataset ROmaiti as frequency difference in MHz over access index a.

The upper plots depict its minimum, mean, and maximum along bit position b for the two devices with artifacts. Access 49 for device D076369 and access 79 for device D080157 clearly show a much higher distance between minimum and maximum than all other accesses. The lower plots depict mean and standard deviation along bit position for all devices. The mean frequency of all ROs on a device within the first few accesses can differ from its mean along accesses by up to 0.5 MHz, whereas the standard deviation of frequency among ROs on the same device remains below 0.05 MHz for all accesses and devices except the two accesses that stood out already in the upper plots. This suggests that the former change in mean frequency is due to a global environmental effect such as die temperature or reference crystal oscillator temperature. The overall mean frequency among accesses, ROs, and devices is 205.1 MHz.

1.9.3. Ahori

Dataset Description

The Ahori dataset contains response data from an arbiter PUF candidate implemented on the Xilinx Virtex-5 FPGA (XC5VLX30) of a SASEBO-GII evaluation board and was published by Hori et al. in connection with [72]. It takes the identifier approach, where 1024 identifiers of 128 bit length are generated on each device one bit at a time, while the arbiter's challenge is generated by a PRNG. This experiment is repeated in 1024 accesses on 45 devices.

Analog response data is not available, because arbiter PUFs produce binary responses intrinsically. The dataset together with MATLAB scripts to apply the metrics from [72] on the data are available for download from [73].

Known Defects

Defects have not been described in [72], but the following have been identified during preparation of this work:

- Quite similar IDs are created on most devices, which impairs unpredictability and is presumably the result of unbalanced propagation delay in the paths towards the arbiter.

- The provided analysis scripts are unable to reproduce the results printed in [72], which means that either the dataset has been modified, the scripts changed, or errors occurred during result preparation for [72].

1.9.4. SRAMxmc

Dataset Description

The SRAMxmc dataset is currently the largest publicly available dataset of COTS SRAM used as PUF candidate and was published online in connection with [**74**]. Two measurement campaigns were performed in 2015 and 2016. Both times, all three SRAMS (program SRAM, first data SRAM, second data SRAM) of an Infineon XMC4500F100K1024 microcontroller, adding up to 160 KiB or 1 310 720 bit, have been read out 101 times with a 5 s power-off between accesses. The second run additionally features simultaneous temperature measurements using the integrated die temperature sensor. To automate data acquisition, a custom readout circuit was employed that allowed a computer program to receive the startup values and temperature measurements sent to it by the microcontroller and power cycle the XMC Relax Lite Kit that housed the microcontroller with well controlled off-time and supply slew-rate. The 2015 run contains 144 devices, whereas the 2016 run contains 143 devices with a small number of devices replaced due to defects in the meantime.

Since SRAM PUF candidates perform quantization to single-bit responses intrinsically, only binary response data is available. It is published online and most easily found by the permanent ID of the corresponding publication, which is 43f781c9642c49e9d6c92dc724d3313261b7cff4. To ensure authenticity and integrity of the data, it is signed with PGP key EF66 6827 EC7A 6C4B 9959 5A08 1EBE 16DD E4E7 C0CC.

Known Defects

Known defects as described in [**74**] are:

- An alternating pattern of 32 bit words with probability for a 1 response among their bits of either 40% or 60%, which presumably is the result of a space saving, two wing layout of the SRAM, where each wing produces every other response word. Thus one of the wings produces responses with a 1 probability of 40% and the other wing produces responses with 60% probability for a 1.

- Fixed response bit positions that turn out the same value on all devices, because the uppermost 128 bit and the lowermost 212 bit of the second SRAM are clobbered by the automatically executed first stage boot loader of the microcontroller.

- HD of the first access to every other access is much larger than HD between other accesses for an unknown reason. Remanence within the SRAM can be ruled out since the behavior remains unchanged if the complement of the read start-up pattern is stored into the SRAM before the next power cycle.

2. Overview of Existing Metrics for Reliability and Unpredictability

As a basis for a thorough discussion and evaluation of existing metrics to identify needs for improvement and extension, this chapter provides an overview on previously proposed or applied metrics for PUFs. As outlined in Sec. 1.7, the focus of this work is on the assessment of the statistical properties of a PUF candidate's CRB. For a more general applicability, this will mostly operate on binary input and output values, because analog response data is only available for certain types of PUF, such as RO PUFs, but not for e.g. SRAM and arbiter PUFs. This is in line with the majority of existing metrics for PUFs. For those cases where analog data is available, Sec. 2.7 gives a brief overview on corresponding metrics to provide additional insight. To ease comparison of existing metrics, this work aims to represent them in a common notation without changing their result or limiting their applicability. Occasionally though, dimensions such as challenges or response bit positions may be added if the original definition did not include them and addition is possible without changing the result for cases where the added dimensions are singleton. An evaluation of selected metrics introduced in this chapter follows in Chpt. 3.

2.1. Bit Position Based Hamming-distance Distributions

To visually inspect histograms of the distributions of intra-class and inter-class HD has been one of the first ways to analyze the reliability and unpredictability of PUFs since at least Lofstrom [5]. The distribution of *intra*-class HD refers to the variation when accessing the same PUF instance with the same challenge, which ideally shows no variation, or an HD of zero, to be able to recognize a previously seen device. An example are the HDs between responses from multiple accesses to the same device with the same challenge, but at different points in time or at different environmental conditions. *Inter*-class HD refers to the HD between responses of separate instances or to different challenges. Their true responses are ideally independent and evenly spaced responses in the response space, because evenly spaced responses maximize for a given response space size and a given number of devices the amount of bit errors during readout that can be tolerated without misidentification. Synonymous terms are "self and others distances" [5] or "like and unlike distributions" [2].

As a string metric, HD counts the number of elements that differ between two strings of arbitrary alphabet. For the case of PUFs though, it typically operates on binary strings, which means it reduces to the sum of digits after a bitwise XOR operation on the two strings. This is usually also the way the HD is written in mathematical equations. To make figures comparable although PUF candidate circuits may have different response lengths, HD is often normalized to response length and reported as FHD or percentage HD.

Section 2.1.1 introduces the formulae and describes typical differences in how to calculate the HDs and how to then create histograms from these samples. Some publications provide further analysis of the distribution of HDs, such as various single value measures of distribution as outlined in Sec. 2.1.4 or to fit an assumed distribution onto the samples as described in Sec. 2.1.2.

2.1.1. Histograms

Histograms provide a visualization of the estimated distribution of HDs and can give a first impression on whether the candidate circuit e.g. has sufficiently few bit errors or whether true responses are evenly spaced in the response space.

Intra-class

The intra-class HDs with regard to the distance of accesses to the corresponding true response can be calculated in this work's notation as

$$\overset{\updownarrow}{h}_{a,c,d,e} = \frac{1}{B}\sum_{b=1}^{B} x_{a,b,c,d,e} \oplus \bar{x}_{b,c,d} \qquad \begin{aligned} a &\in \{1,\dots,A\},\\ c &\in \{1,\dots,C\},\\ d &\in \{1,\dots,D\},\\ e &\in \{1,\dots,E\}, \end{aligned} \tag{2.1}$$

where the arrow indicates that this is a collection of samples, and the vertical direction represents intra-class. Histograms of solely the intra-class HDs can be found in e.g. [2, 17, 32, 70]. The histograms are typically built over the dimensions a, c, d, which provides an impression of the overall distribution, but does not allow to compare e.g. the behavior of individual devices with each other.

To define the true response $\bar{x}_{b,c,d}$, several different approaches exist: In many cases it has been defined as the first response at reference environmental conditions in the dataset, cf. e.g. [2, 5, 12, 15, 17, 31, 75], [76]. Maiti et al. [70] defined it as the result of bit extraction on the mean of the analog response data among all accesses. Ignatenko et al. [77] chose the 12th out of 25 measurements, all at reference environmental conditions, as true response. Hori et al. [72] and presumably also Su et al. [67] used a majority vote among binary responses from all accesses in the dataset. This is performed individually per response bit position, e.g. 0011 if three accesses return 0011, 0111, 0001. It ensures that for each individual bit position more accesses match the true response than not, as long as the majority vote and the analysis are performed on the same dataset. Holcomb et al. [66, 78] obtained true responses by majority vote from another dataset, which does not provide this guarantee.

Such histograms typically covered accesses during reference environmental conditions only, even where data for other environmental conditions were measured. The effect of environmental conditions on the intra-class HD distribution has instead been given through single value measures of distribution, cf. Sec. 2.1.4, or in line plots, for example in [23, 76] over temperature, in [17] to analyze SRAM remanence and aging, in [6] over supply voltage, in [67, 70, 75] over supply voltage and temperature, and in [75] also over supply voltage ramp up time. An exception is Suh and Devadas [32], who presented a histogram of the intra-class HD distribution for the highest temperature, highest voltage test case with the true response from reference conditions, and Holcomb et al. [78], who provided an overlayed histogram for three temperatures with the true response from reference conditions. Su et al. [67] provided a table of intra-class HDs after accelerated aging. Böhm and Hofer [79] listed intra-class HDs under various environmental conditions individually, presumably because only one access was made under each condition.

Inter-class

For the inter-class HDs exists no obvious reference, such as the true response for the intra-class HDs, therefore the distribution is typically built over every possible pair in the dataset. What constitutes a pair, differs in multiple ways, though. The first option is to either compare only the true response of each device or the responses from all accesses. The second option is whether challenges are handled individually or are mixed into the pair building.

If only true responses are compared and challenges are analyzed individually, the collection of samples is expressed in this work's notation as

$$\overset{\leftrightarrow}{h}_{c,d''} = \frac{1}{B}\sum_{b=1}^{B} \bar{x}_{b,c,d} \oplus \bar{x}_{b,c,d'} \qquad \begin{aligned} c &\in \{1,\dots,C\},\\ d &\in \{1,\dots,D-1\},\\ d' &\in \{d+1,\dots,D\},\\ d'' &\in \{1,\dots,\tfrac{D(D-1)}{2}\}, \end{aligned} \tag{2.2}$$

where the horizontal direction of the arrow represents inter-class. Note that although d'' uses the index for devices, it is rather a device-pair index. Pappu [2] instead built pairs on the grounds of devices *and* challenges, so every true response $\underline{\bar{x}}_{c,d}$ is compared to every other true response $\underline{\bar{x}}_{c',d'}$ as long as $c \neq c'$ or $d \neq d'$:

$$\overset{\leftrightarrow}{h}_{g''} = \frac{1}{B} \sum_{b=1}^{B} \bar{x}_{b,c,d} \oplus \bar{x}_{b,c',d'} \qquad \begin{gathered} d \in \{1, \ldots, D\}, \\ c \in \begin{cases} \{1, \ldots, C\} & d < D \\ \{1, \ldots, C-1\} & d = D \end{cases}, \\ d' \in \begin{cases} \{d, \ldots, D\} & c < C \\ \{d+1, \ldots, D\} & c = C \end{cases}, \\ c' \in \begin{cases} \{c+1, \ldots, C\} & d' = d \\ \{1, \ldots, C\} & d' > d \end{cases}, \\ g'' \in \{1, \ldots, \tfrac{CD(CD-1)}{2}\}. \end{gathered} \tag{2.3}$$

For single-challenge PUFs, both approaches coincide. Histograms to visualize solely the inter-class HD distribution in either notion can be found in e.g. [2, 6, 17, 32, 57, 67, 70, 75]. Note that Suh and Devadas used the term “inter-chip variation” [32] for this collection of samples, although the name had been used for different metrics by e.g. [12, 31].

Holcomb et al. [66, 78] chose to compare the response from all accesses to every device to the true response of all other devices. Since they focus on single-challenge PUFs, index c is omitted for brevity, so the resulting collection of samples follows in this work’s notation from

$$\overset{\leftrightarrow}{h}_{a,d'',e} = \frac{1}{B} \sum_{b=1}^{B} x_{a,b,d,e} \oplus \bar{x}_{b,d'} \qquad \begin{gathered} a \in \{1, \ldots, A\}, \\ d \in \{1, \ldots, D-1\}, \\ d' \in \{d+1, \ldots, D\}, \\ d'' \in \{1, \ldots, \tfrac{D(D-1)}{2}\}, \\ e \in \{1, \ldots, E\}, \end{gathered} \tag{2.4}$$

where a histogram covers both a, d'', but is depicted separately for each e if data for multiple environmental conditions is available. Although Holcomb et al. prefer to compare the responses from all accesses, they additionally report the distribution according to (2.2) in [66], but not in [78].

Instead of one true response at reference environmental conditions, Katzenbeisser et al. [15] defined individual true responses for each environmental condition, so the collection of samples follows from

$$\overset{\leftrightarrow}{h}_{c,d'',e} = \frac{1}{B}\sum_{b=1}^{B} \bar{x}_{b,c,d,e} \oplus \bar{x}_{b,c,d',e} \qquad \begin{aligned} c &\in \{1,\dots,C\}, \\ d &\in \{1,\dots,D-1\}, \\ d' &\in \{d+1,\dots,D\}, \\ d'' &\in \{1,\dots,\tfrac{D(D-1)}{2}\}, \\ e &\in \{1,\dots,E\}. \end{aligned} \tag{2.5}$$

The distribution was reported in individual histograms per environmental condition. However, it was not stated whether the histograms cover both remaining dimensions, device-pairs and challenges, or one of both dimensions was processed e.g. by a mean operation, before creating the histogram.

Combined

Some authors have chosen to report both intra-class and inter-class HD distribution in a combined histogram, so the gap or overlap between the distributions becomes more obvious. The motivation for this kind of visualization is rarely stated explicitly, but where it was given, it was claimed that, if no overlap between both distributions is visible, correct identification for all devices is achievable, i.e. no device produces responses due to noise that are closer to the true response of another device than its own true response. However, as further discussed in Sec. 3.2, this depends on whether the depicted inter-class HD distribution is built upon all accesses as in (2.4) or true responses only, as in (2.2), (2.3). Histograms of the former type are given in e.g. [2], and of the latter type in e.g. [66, 78], but there are also cases where the type remains unclear, such as [5, 75, 80].

Note that very similar looking histograms were sometimes provided that depicted different data, though. An example is Su et al. [67], who plotted the inter-class distribution based on true responses, but seem to preprocess the intra-class distribution to a single value per device, given the granularity of the bar heights and the name “measured unstable bits” [67]. Maes et al. [34] built the combined histogram of intra- and inter-class from the 49 bit *sub*-responses instead of the entire response.

2.1.2. Theoretical Distribution Fits

To compare experiment data with theoretical models or assumptions, the intra-class or inter-class HD histograms have sometimes been overlayed with probability distributions such as a normal distribution or a binomial distribution. This was already done by Lofstrom [5], but without giving the type and parameters of the overlayed distribution. Normal distribution fits can be found in e.g. [17, 67] for both inter-class and intra-class HD distribution, or [6] for the inter-class HD distribution. Binomial distribution fits were presented in e.g. [32, 34] for inter-class and intra-class HD distribution.

The conformance of the inter-class HD distribution to a normal distribution can serve as a metric according to Kömürcü and Dündar [81]. However, they did not state exactly how to calculate this metric, but merely describe that it is "calculated via correlating the HD's [sic] of PUF data with the ideal Gaussian distribution" [81].

A more elaborate example was given by Pappu [2], where the univariate Bernoulli approach with $p = 0.5$ was taken and which dealt with 2 400 bit responses produced from images of the speckle patterns through a "Gabor hash algorithm" [2]. The corresponding binomial distribution with parameters $n = 2400$ and $p = 0.5$, denoted as $\mathfrak{B}\left(2400, 0.5\right)$, was compared to the inter-class distribution, which revealed that the bell-shaped curve of the binomial distribution was much wider than the distribution of the experimental data. The conclusion drawn from this finding was that the algorithm does not produce entirely independent response bits and based on the variance of the binomial distribution, $np(1-p)$, the number of independent unbiased response bits was computed as 228. A $\mathfrak{B}\left(228, 0.5\right)$ distribution thus fitted well to the experimental inter-class distribution. In the same way the intra-class distribution was estimated to follow a $\mathfrak{B}\left(41, 0.2525\right)$ distribution, but this matched noticeably less to the experimental data than for the inter-class case.

2.1.3. Tables And Lists

Publications with a low number of devices and accesses and with a single-challenge PUF – or a multi-challenge PUF with fixed challenge – sometimes presented one or both distributions in a matrix-like table. Pappu [2] presented both, where the main diagonal of the table contained the intra-class FHD for each device individually, and the upper and lower triangles showed the inter-class distribution. Although FHD is a commutative operation, so both triangles should be symmetric with respect to the main diagonal, this is not the case in [2] and no further explanation for this is given.

Su et al. [67] used a similar matrix-like table. The upper and lower triangles also differ in this case, but that is because they refer to two different layout techniques of the ASIC implementation. The main diagonal was filled with zeros rather than depicting the intra-class variation.

Böhm et al. [79] provided a list of HDs between the three devices analyzed in their work.

2.1.4. Single Value Measures of Distribution

In addition to or instead of histograms that visualize the distribution of intra-class and inter-class HD distribution, many publications have included numerical figures such as the minimum inter-class HD, mean inter-class HD, or maximum intra-class HD. While histograms have often been given for reference conditions, numerical values were used to state the effect of changes over environmental conditions, including aging. Most publications refrained from the definition of a mathematical symbol, but explained the conditions of calculation textually.

The following list starts with measures on the intra-class distribution, followed by measures on the inter-class distribution, and in the end measures for both.

Intra-Class Maximum

In this work's notation, environmental conditions are reflected by index e, thus

$$\overset{\ulcorner}{h}_e = \max_{a,c,d} \overset{\updownarrow}{h}_{a,c,d,e} \tag{2.6}$$

provides the maximum intra-class HD under each environmental condition the dataset contains, individually, with regard to the true response under reference conditions. The accent visualizes the intra-class maximum intuitively as a vertical line for intra-class and a bar at its upper end to represent the maximum. The metric may be used as an estimate for the maximum number of bit errors the post-processing needs to correct if a sufficient number of devices is taken into account. It can also be used to evaluate the effect of artificial aging as reported by Lofstrom [5] or temperature as reported by Schrijen et al. [76]. Extension to report the maximum intra-class HD over all environmental conditions is trivial by moving e from an index to the $\max$ operator. The $\max$ operator can also be applied on arbitrary subsets of e, as done for example by Guajardo et al. [17], who reported the maximum intra-class HD among multiple temperatures, but separately for reference conditions and aging. Without data for multiple environmental conditions, the approaches coincide and there is only one number, as reported in e.g. [66, 80]. Maiti et al. [70] reported the maximum and minimum intra-class HD under reference environmental conditions and the maximum over all environmental conditions.

Katzenbeisser et al. [15] restricted the $\max$ operator to accesses and challenges

$$\overset{\ulcorner}{h}_{d,e} = \max_{a,c} \overset{\updownarrow}{h}_{a,c,d,e} \tag{2.7}$$

and constructed a histogram among devices for each environmental condition and PUF type.

Kömürcü and Dündar [81] restricted the max operator to accesses only and focussed on single-challenge PUFs, so challenges c were ignored,

$$\overset{\ulcorner}{h}_{d,e} = \max_{a} \overset{\updownarrow}{h}_{a,d,e} \tag{2.8}$$

and reported the results in a table. They furthermore suggested to use the relative improvement in $\overset{\ulcorner}{h}_{d,e}$ “after masking the most erroneous 3 bits” [81] and “after majority voting for 3 times” [81] as additional metrics.

Intra-Class Mean

In a similar way, the mean intra-class HD at given environmental conditions

$$\overset{\shortparallel}{h}_{e} = \frac{1}{ACD} \sum_{a=1}^{A} \sum_{c=1}^{C} \sum_{d=1}^{D} \overset{\updownarrow}{h}_{a,c,d,e} \tag{2.9}$$

is denoted by the vertical direction for intra-class and a parallel bar to represent the mean. It may be used to estimate the usual number of bit errors to correct by post-processing e.g. in the key storage scenario. For the case of single-challenge PUFs, i.e. with c omitted, it has been named “Reliability” by Maiti et al. [70]. Reports of this figure can be found in e.g. [2, 5], [6, 17, 57, 70, 76]. Some authors have reported an average intra-class HD, which only presumably refers to the mean, for example in [32, 67].

Confidence intervals (CIs) for the intra-class mean, though without providing the underlying statistical model or any other explanation of how they were calculated, were given for example by Su et al. [67].

Kömürcü and Dündar [81] focussed on single-challenge PUFs, so c is omitted, and calculated the mean only per device

$$\overset{\shortparallel}{h}_{d,e} = \frac{1}{A} \sum_{a=1}^{A} \overset{\updownarrow}{h}_{a,d,e}. \tag{2.10}$$

While this provided one value per device, they reported only one value per PUF design in a table without explanation how the values per device were processed into a single number.

Intra-Class Median

A rarely used, but statistically more robust figure than the mean intra-class HD $\overset{\shortparallel}{h}_{e}$ is the median intra-class HD $\overset{\vdash}{h}_{e}$, i.e. the middle element after sorting the intra-class HDs by value respectively the mean of the two middle values in case of an even number of samples. It is reported for example by Pappu [2].

Average Intra-Class without True Responses

A metric that evaluates the intra-class HD distribution without explicit definition of a true response is “Reliability” as defined by Merli et al. [33]. It has been defined as the mean of all possible pairs of accesses on a device similar to the inter-class mean, which takes the mean of all possible pairs of devices with their true response. Targeted at single-challenge PUFs, it does not consider challenges, so c is omitted. According to [33], the environmental conditions may or may not be varied during accesses, which results in two possible notations:

$$\overset{\text{Mr}}{h}_{e} = 1 - \frac{2}{A(A-1)BD} \sum_{d=1}^{D} \sum_{a=1}^{A-1} \sum_{a'=a+1}^{A} \sum_{b=1}^{B} x_{a,b,d,e} \oplus x_{a',b,d,e} \tag{2.11}$$

$$\overset{\text{Mr}}{h} = 1 - \frac{2}{AE(AE-1)BD} \sum_{d=1}^{D} \sum_{e=1}^{E} \dots$$
$$\dots \sum_{a=1}^{\substack{A \text{ if } e<E \\ A-1 \text{ if } e=E}} \; \sum_{e'=\begin{cases} e \text{ if } a<A \\ e+1 \text{ if } a=A \end{cases}}^{E} \; \sum_{a'=\begin{cases} a+1 \text{ if } e'=e \\ 1 \text{ if } e'>e \end{cases}}^{A} \; \sum_{b=1}^{B} x_{a,b,d,e} \oplus x_{a',b,d,e'} \tag{2.12}$$

So the HD between all pairs of accesses, either at a certain environmental condition or all conditions, followed by a mean among devices.

Intra-Class Variance and Standard Deviation

To report the spread of the intra-class HD distribution, its variance was given for example by Pappu [2]. Guajardo et al. [17] instead reported the standard deviation, which provides the same information.

Intra-Class Minimum

At first glance, intra-class minimum may be considered irrelevant for most use cases, because both recognition of a response for cryptoless authentication as well as error correction for key storage can handle from zero up to a certain limit of bit errors. However, it can be used as an additional point of support in the fit of an assumed distribution, for example as done by Schrijen et al. [76].

Inter-Class Minimum

The true responses relevant for the inter-class distribution are typically defined at reference environmental conditions. Although there are still caveats such as the previously mentioned definition of pairs of devices vs. pairs of challenges and devices, this leaves less options for misunderstanding. A commonly reported figure is the minimum inter-class HD

$$\overset{\llcorner}{h}_c = \min_{d''}\left(\overset{\leftrightarrow}{h}_{c,d''}\right), \text{ respectively} \tag{2.13}$$

$$\overset{\llcorner}{h} = \min_{g''}\left(\overset{\leftrightarrow}{h}_{g''}\right), \tag{2.14}$$

where d'' is the device pair index introduced in (2.2) and g'' is the device-and-challenge pair index introduced in (2.3). The accent is a horizontal line to denote inter-class with a bar on its left end to represent a minimum. The figure may be used in combination with the maximum intra-class HD to infer a border on the number of bit errors for both authentication and key storage. If more bit errors than the inter-class minimum are accepted during authentication or corrected in a key-storage scenario, some devices can impersonate each other, which is undesirable. Examples of this figure can be found in e.g. [5, 66, 70].

Kömürcü and Dündar [81] suggested to use a metric that further processes the minimum inter-class HD. It is allegedly based on the Gilbert-Varshamov bound (GVB) and relates the Shannon entropy of the minimum FHD, $\overset{\llcorner}{h}\log_2(\overset{\llcorner}{h}) + (1 - \overset{\llcorner}{h})\log_2(1 - \overset{\llcorner}{h})$, to the logarithm of "the number of circuits" [81] that obviously refers to the number of devices. In this work's notation it becomes

$$\overset{\text{Ku}}{r} = \frac{\frac{\log_2(D)}{B}}{1 + \overset{\llcorner}{h}\log_2(\overset{\llcorner}{h}) + (1 - \overset{\llcorner}{h})\log_2(1 - \overset{\llcorner}{h})}. \tag{2.15}$$

Targeted at single-challenge PUFs, it does not consider challenges.

Inter-Class Mean (Uniqueness)

The most common figure to report is the mean inter-class HD

$$\overset{=}{h}_c = \frac{2}{D(D-1)}\sum_{d''=1}^{D(D-1)/2}\left(\overset{\leftrightarrow}{h}_{c,d''}\right), \text{ respectively} \tag{2.16}$$

$$\overset{=}{h} = \frac{2}{CD(CD-1)}\sum_{g''}^{CD(CD-1)/2}\left(\overset{\leftrightarrow}{h}_{g''}\right), \tag{2.17}$$

which coincide for single-challenge PUFs into

$$\overset{=}{h} = \frac{2}{D(D-1)} \sum_{d''=1}^{D(D-1)/2} \left(\overset{\leftrightarrow}{h}_{d''} \right) . \tag{2.18}$$

Consistent with the intra-class mean, it is denoted by two parallel bars, but in horizontal rather than vertical direction. Equation (2.18), which may also be used for multi-challenge PUFs with fixed challenge, has been named “Uniqueness” by Maiti et al. [56]. The measure is frequently used as an overall quality indicator for the unpredictability of a PUF candidate circuit and further discussed in Sec. 3.6. Reports on mean inter-class HD can be found in e.g. [2, 6, 56], [57, 67, 70, 81]. Some authors reported an average inter-class HD, which only presumably refers to the mean, for example Suh and Devadas [32].

CIs for the inter-class mean, though without providing the underlying statistical model or any other explanation of how they were calculated, were given by Su et al. [67].

Inter-Class Maximum

The maximum inter-class HD

$$\overset{\lrcorner}{h}_c = \max_{d''} \left(\overset{\leftrightarrow}{h}_{c,d''} \right) , \text{ respectively} \tag{2.19}$$

$$\overset{\lrcorner}{h} = \max_{g''} \left(\overset{\leftrightarrow}{h}_{g''} \right) , \tag{2.20}$$

where the accent has its bar on the right end to indicate maximum rather than on the left end to indicate minimum, is rarely used. An example can be found in Maiti et al. [70].

Inter-Class Median, Variance, Standard Deviation

As for the intra-class HD distribution, median, variance, and standard deviation of the inter-class HDs are rarely reported. An example, however, is Pappu [2], who reported the inter-class median and variance.

Quantiles

Occasionally, arbitrary quantiles of the intra-class or inter-class HD distributions were reported. An example is Holcomb et al. [66], where among others it was reported that four out of 300 intra-class HDs were larger than 425 bit, i.e. the 296/300-quantil is 425 bit.

Estimation Based on Fitted Distribution

Sometimes the mean, variance, or standard deviation were not reported for the HD distribution itself, but for a standard distribution fitted to the observed data. Examples of this for the inter-class distribution can be found in [17, 76].

Average Inter-Class Without True Responses

In a similar way, Merli et al. [33] defined a way to calculate the average inter-class HD distribution without true responses, again termed "Uniqueness". For this, all accesses of all possible device pairs were compared:

$$\overset{\text{Mu}}{h}_{e} = \frac{2}{A^2BD(D-1)} \sum_{d=1}^{D-1} \sum_{d'=d+1}^{D} \sum_{a=1}^{A} \sum_{a'=1}^{A} \sum_{b=1}^{B} x_{a,b,d,e} \oplus x_{a',b,d',e} \tag{2.21}$$

2.2. Challenge Based Hamming-distance Distributions

As HD is a string metric and PUF data is multi-dimensional, it is a relevant question which dimension is considered to establish the string. The previous section assumed that this is the dimension of response bit positions. However, in particular for multi-challenge PUFs where a device produces a single bit of response per challenge, it can be a valid choice to define the string along the dimension of challenges as performed for example by Gassend et al. [23] or Kalyanaraman et al. [27]. Apart from this change in dimension for strings, the approach is similar to that described in the previous section. Therefore this section likewise starts with a desciption of how to obtain the HDs, followed by examples of previous work that presents histograms and single values measures of distribution. Note that the choice to build strings along the dimension of challenges is less frequently used than strings along response bit positions, so the number of examples is scaled accordingly.

The samples to estimate the intra-class HD distribution follow in this work's notation from

$$\overset{\updownarrow}{g}_{a,d,e} = \frac{1}{C} \sum_{c=1}^{C} x_{a,c,d,e} \oplus \bar{x}_{c,d} \qquad \begin{array}{l} a \in \{1,\dots,A\}, \\ d \in \{1,\dots,D\}, \\ e \in \{1,\dots,E\}. \end{array} \tag{2.22}$$

In a similar way, the inter-class HDs may be given as

$$\overset{\leftrightarrow}{g}_{d''} = \frac{1}{C}\sum_{c=1}^{C} \bar{x}_{c,d} \oplus \bar{x}_{c,d'} \qquad \begin{array}{c} d \in \{1,\ldots,D\}, \\ d' \in \{d+1,\ldots,D\}, \\ d'' \in \{1,\ldots,\frac{D(D-1)}{2}\}. \end{array} \tag{2.23}$$

Although not explicitly stated, it may be concluded from textual description that the term "interchip variation" used by Lim et al. in [31] refers to the inter-class distribution, although the term was previously introduced for a single value measure by Lee et al [12], as described below.

For multi-challenge PUFs that provide multiple bits of response, the HD between response bit positions on individual devices can give a hint on dependencies among the response bit positions. According to textual description by Majzoobi et al. [64], this kind of inter-class HD distribution follows from

$$\overset{\leftrightarrow}{g}_{b'',d} = \frac{1}{C}\sum_{c=1}^{C} \bar{x}_{b,c,d} \oplus \bar{x}_{b',c,d} \qquad \begin{array}{c} b \in \{1,\ldots,B\}, \\ b' \in \{b+1,\ldots,B\}, \\ b'' \in \{1,\ldots,\frac{B(B-1)}{2}\}, \\ d \in \{1,\ldots,D\}. \end{array} \tag{2.24}$$

and is reported in [64] as heatmap for a device.

2.2.1. Histograms

Histograms based on the HDs calculated by above formulae are used to provide a first visual impression of the distribution, the same way as for bit position based HDs. An example for the inter-class distribution is given by Lim et al. [31], which additionally includes an overlay of a bell shaped, but not further specified standard distribution, kernel estimate, or something similar. Kalyanaraman et al. [27] provide the combined histograms familiar from the previous section, which contain both inter-class and intra-class distribution to visualize the separability of both.

2.2.2. Single Value Measures of Distribution

Challenge Based Intra-class Distribution

An example for typical single value measures is intra-class mean, which was reported for example by Kalyanaraman et al. [27].

Lee et al. [12] and Lim et al. [31] used a metric termed "noise" that is textually defined to be "the probability that a newly measured response is different from the corresponding [true] response” [12], but neither work provides a mathematical formula. Since they assume the measure to hold for all devices and only to vary by environmental conditions such as temperature and supply voltage, one may conclude that it is an average among accesses and devices of $\overset{\updownarrow}{g}_{a,d,e}$. Due to the remaining uncertainty in exact definition, the measure is denoted as $\overset{\text{Ln}}{p}_{e}$ in this work. It is used as parameter to calculate the probability of misidentification according to Lee et al., cf. Sec. 2.5.1.

Challenge Based Inter-class Distribution

Examples for typical single value measures are inter-class mean, reported for example by Kalyanaraman et al. [27], and inter-class minimum, reported for example by Lim et al. [31]. Furthermore, the minimum, average, and maximum are reported by Lim et al. in a rarely found experiment to investigate how the inter-class HD differs between “die-to-die, wafer-to-wafer, and lot-to-lot” [31], based on the inter-class HD distribution among devices from two individual wafers and both wafers combined, in a bar chart.

Gassend et al. [23] reported an average of $\overset{\leftrightarrow}{g}_{d''}$, but without specifying whether it refers to the mean, median, etc. and whether it is based on all possible pairs of devices or e.g. one device compared to all others. While it is neither explicitly stated nor any formula given, it can be concluded that the term “interchip variation” as used by Lee et al. [12] refers to the same or at least a similar measure. Lee et al. originally defined the term as the “probability that the first measured [i.e. true in this work’s notation] response for a given challenge on a first chip is different from the first measured [i.e. true] response for the same challenge on a second chip” [12], but since it is used in [12] without further index, it would need to hold among all challenges and pairs of devices, which suggests above conclusion. Due to the remaining uncertainty in definition, it is denoted in this work as $\overset{\text{Lv}}{p}$. The measure is used in the calculation of the probability of misidentification according to Lee et al., cf. Sec. 2.5.1.

2.3. Identifier Based Hamming-distance Distributions

Yet another way to built binary strings out of PUF data is to combine the bit positions of responses to multiple different challenges into identifiers and compute the HDs between these identifiers. As introduced in Sec. 1.8, metrics that follow this approach contain indices k and l rather than b and c, where the mapping from the former to the latter indices is implementation defined. Still, the following equation typically holds:

$$BC = KL \tag{2.25}$$

Common examples for this approach are the metrics by Hori et al. [72] and those in Maes' PhD thesis [82]. Similar to the previously described approaches for bit position based HDs and challenge based HDs, the distribution of HDs between identifiers can be described either by histograms, single value measures of distribution, or a fit of a standard distribution.

Maes [82] compared both single-challenge and multi-challenge PUF designs against each other. In an attempt to provide a fair comparison, identifiers were either chosen to equal the response of the single-challenge design, e.g. an SRAM PUF that produces a 64 Kibit identifier, or a concatenation of responses from all instances of the multi-challenge design on the device for multiple challenges, e.g. a 64 Kibit identifier from 256 arbiter PUFs evaluated for 256 different challenges or a 32 Kibit identifier from 128 XOR-arbiter PUFs for the same number of challenges. Thus one identifier per device was created, so $K = 1$ and $L = BC$ in this work's notation.

Regarding single values measures of distribution, Maes [82] reported the mean and standard deviation for inter-class and intra-class distribution. Additionally, the 1st percentile and minimum for the inter-class distribution as well as the 99th percentile and maximum for the intra-class distribution were given. The percentiles and minimum respectively maximum were supposed to give a better view "of the left [respectively right] tail of the distribution, which [are] of interest when quantifying identifiability" [82]. As further discussed in Sec. 3.2, an overlap between the left tail of the inter-class distribution and the right tail of the intra-class distribution may lead to misidentification under certain circumstances.

Maes [82] also used standard distributions as a description. Instead of the usual approach to fit the standard distribution's PDF to the histogram, though, Maes chose the parameters of the standard distribution according to the single value measures of distribution. More precise, binomial distributions for inter-class and intra-class distribution were selected so an experiment with this distribution, and of the same size as the measurement population, would reproduce the observed mean, percentile, and minimum respectively maximum, with sufficient probability.

Hori et al. [72] defined a set of metrics based on the identifier approach. Three out of five metrics – named *Correctness*, *Diffuseness*, and *Uniqueness* – have been based on HDs with various amounts of averaging and normalization to achieve a consistent scale from 0 (worst) to 1 (best). Their work used an arbiter PUF as example, so $B = 1$ thus $C = KL$.

Correctness according to [72] has been defined as

$$\overset{\text{Hc}}{g}_{d} = 1 - \frac{2}{AKL} \sum_{l=1}^{L} \sum_{k=1}^{K} \sum_{a=1}^{A} (x_{a,l,k,d} \oplus \bar{x}_{l,k,d}), \tag{2.26}$$

where the true identifier $x_{l,k,d}$ was found by majority vote among all samples. The metric is supposed to measure the amount of PUF cells that went defect after the true identifier had been determined, although this can only work if the true identifier was *not* found by majority vote among samples of *this* dataset, but defined otherwise, e.g. through a previous dataset. For a more detailed discussion see Sec. 3.3.4. Note furthermore that the metrics in [72] have not been designed to consider environmental conditions, so index e is omitted.

Diffuseness has been defined as

$$\overset{\text{Hd}}{g}_{d} = \frac{4}{LK^2} \sum_{k=1}^{K-1} \sum_{k'=k+1}^{K} \sum_{l=1}^{L} (\bar{x}_{l,k,d} \oplus \bar{x}_{l,k',d}), \tag{2.27}$$

and therefore requires multiple identifiers per device, i.e. $K > 1$. Its purpose is to measure how unpredictable the identifiers produced by the same device are.

Uniqueness[1] has been defined in a per-device and an overall form and is the only metric by Hori et al. that compares between devices. The version for an individual device compared to the other devices in the dataset,

$$\overset{\text{Hu}}{g}_{d} = \frac{2}{LKD} \sum_{k=1}^{K} \sum_{l=1}^{L} \sum_{d'=1, d' \neq d}^{D} (\bar{x}_{l,k,d} \oplus \bar{x}_{l,k,d'}), \tag{2.28}$$

had a nominator of 4 in [72], but since its mean among devices,

$$\overset{\underline{\text{Hu}}}{g} = \frac{1}{D} \sum_{d=1}^{D} \overset{\text{Hu}}{g}_{d}, \tag{2.29}$$

shall be equal to the version of Uniqueness that measures the entire dataset,

$$\overset{\underline{\text{Hu}}}{g} = \frac{4}{LKD^2} \sum_{k=1}^{K} \sum_{l=1}^{L} \sum_{d=1}^{D-1} \sum_{d'=d+1}^{D} (\bar{x}_{l,k,d} \oplus \bar{x}_{l,k,d'}), \tag{2.30}$$

the nominator in either (2.28) or (2.30) has to be corrected due to different indexing. Observe that (2.28) compares device d against all other devices, whereas (2.30)

[1]Note the name conflict with Uniqueness according to Maiti et al. [70] and Merli et al. [33].

compares it only against devices with larger index. Which of (2.28), (2.30) is to be corrected can be resolved by considering that all metrics by Hori et al. are supposed to produce unity for the optimal case. This allows to conclude that the nominator in (2.28) needs to be 2 instead of 4. Note that Uniqueness for individual devices may still produce values larger than 1 and up to 2. Consider for this a dataset with all but a single device featuring an all-0 response, while one distinct device produces an all-1 response. $\overset{\text{Hu}}{g}_d$ for the distinct device will then reach $^2/_D(D-1) = 2 - {^2/_D}$, which approaches 2 for $D \to \infty$.

Note that indices k and l are used interchangeably in Correctness and Uniqueness. Their result is thus independent of how C is divided into K and L and one may substitute $b \geq 1$ and c for l and k. The similarities of Correctness and Uniqueness with the metrics by Maiti et al. are discussed in Sec. 3.3.

CIs for the mean of all three metrics were provided by use of the central limit theorem (CLT), which according to [72] implies that the results of all three metrics follow a normal distribution among all devices in the dataset. The CI for their mean may then be calculated from the percentiles of a t-distribution with $D-1$ degrees of freedom [72].

2.4. Hamming-weight Figures

2.4.1. Inter-Class Mean

In addition to HDs, Hamming-weights (HWs) have been a popular tool to analyze the performance of PUFs. They are also easy to use, since they equal a simple arithmetic mean for binary data represented as 0 and 1. However, since PUF data is multi-dimensional, there remains a degree of freedom regarding which dimensions to include in the mean and which not. All eight possible combinations of response bit positions b, challenges c, and devices d can be found in the literature, where selection depends on single- vs. multi-challenge PUF and the statistical defect to look for.

One such defect would be a response bit position that has a bias towards either value, as this would allow an attacker to make an educated guess for the response bit at this position. Since dimension b needs to remain for this, one may take the mean among challenges and devices

$$m_b = \frac{1}{CD} \sum_{c=1}^{C} \sum_{d=1}^{D} \bar{x}_{b,c,d} \tag{2.31}$$

as reported for example by Pappu [2], or check the bias also for individual challenges

$$m_{b,c} = \frac{1}{D} \sum_{d=1}^{D} \bar{x}_{b,c,d} \tag{2.32}$$

as reported for example by Gassend et al. [23]. For single-challenge PUFs both approaches coincide, which yields "Bit-Alias"

$$m_{b} = \frac{1}{D} \sum_{d=1}^{D} \bar{x}_{b,d} \tag{2.33}$$

as defined by Maiti et al. [70]. Because the number of values to report is often rather high, in particular for (2.32), the values are typically plotted as a line plot [2, 70], a histogram [23, 57], or a heatmap that reflects the location of the corresponding PUF cell on the die [6, 67]. The latter has the additional benefit that spatial artifacts such as an increased probability for 1 as outcome at the edges of the circuit are very prominently visualized. Note that this does not include spatial correlations, though, cf. Sec. 4.2.2 for methods to detect those. Some authors, e.g. Maiti et al. [70] reported also the minimum and maximum among positions.

Another defect can be devices that tend to either value with their entire response, i.e. with all their response bit positions, because it allows an educated guess for *every* response bit position and the number of possible responses with a certain HW decreases strongly for bias towards any direction. A check for this requires dimension b to be part of the mean and is thus orthogonal to the previous paragraph. Still, it may again be calculated for individual challenges as

$$m_{c,d} = \frac{1}{B} \sum_{b=1}^{B} \bar{x}_{b,c,d} \tag{2.34}$$

or combined over bit positions and challenges as

$$m_{d} = \frac{1}{BC} \sum_{b=1}^{B} \sum_{c=1}^{C} \bar{x}_{b,c,d}, \tag{2.35}$$

which coincides for single-challenge PUFs to

$$m_{d} = \frac{1}{B} \sum_{b=1}^{B} \bar{x}_{b,d} \tag{2.36}$$

and was given the name "Uniformity" in [70]. To report the distribution, a line plot was used in e.g. [57, 70]. Its minimum, average and maximum among devices for a single-challenge PUF was reported for example by Maiti et al. [70]. Böhm and Hofer [79] chose to list the values of their experiment. Katzenbeisser et al. [15] used separate true responses for each environmental condition, which requires (2.35) to be extend to

$$m_{d,e} = \frac{1}{BC} \sum_{c=1}^{C} \sum_{b=1}^{B} \bar{x}_{b,c,d,e}. \tag{2.37}$$

Note that they did not provide a formula, but textually described to calculate the "average", which is assumed to refer to the mean. The results were reported as histograms over devices for individual environmental conditions. In a similar approach, Schrijen et al. [76] reported for each environmental condition e individually the minimum, mean, and maximum of

$$m_{a,d,e} = \frac{1}{B} \sum_{b=1}^{B} x_{a,b,d,e} \tag{2.38}$$

among accesses a and devices d, without selection of a true response. Their results were tabulated.

For multi-challenge PUFs, two further metrics are reasonable. First, a test for response bit positions of individual devices to be biased among challenges

$$m_{b,d} = \frac{1}{C} \sum_{c=1}^{C} \bar{x}_{b,c,d}, \tag{2.39}$$

which was reported for example by Majzoobi et al. [64] in a bar chart and by Kalyanaraman et al. [27] in a histogram. The latter referred to this metric as "Uniformity" [27], which created another name conflict with Uniformity as defined earlier by Maiti et al. [70]. The second metric for multi-challenge PUFs is a test for challenges that tend to produce the same response in all response bit positions and on all devices:

$$m_c = \frac{1}{BD} \sum_{b=1}^{B} \sum_{d=1}^{D} \bar{x}_{b,c,d} \tag{2.40}$$

Finally, a global dataset bias that incorporates all dimensions can be defined as

$$m = \frac{1}{BD} \sum_{b=1}^{B} \sum_{d=1}^{D} \bar{x}_{b,d} \tag{2.41}$$

for single-challenge PUFs, respectively

$$m = \frac{1}{BCD} \sum_{b=1}^{B} \sum_{c=1}^{C} \sum_{d=1}^{D} \bar{x}_{b,c,d} \tag{2.42}$$

for multi-challenge PUFs. Examples can be found in [2, 67, 70], where Su et al. [67] also contains CIs, though without further explanation as to how they were calculated. Additionally, regional biases can be calculated by limiting b to a subset of all response bit positions in above formulae. For example, Su et al. [67] reported the mean number of 1s for each *column* of their array individually in a line plot.

2.4.2. Intra-Class Mean

To investigate the intra-class distribution for individual response bit positions, the number of times they are set among accesses can be a metric:

$$m_{b,c,d,e} = \frac{1}{A} \sum_{a=1}^{A} x_{a,b,c,d,e} \tag{2.43}$$

An ideal PUF is free of noise and reports always the true response, but since the true response for a given bit position of a particular device may be 0 or 1, either value is an optimum for this metric, while 0.5 indicates the PUF is rather a random number generator (RNG). The metric is reported for example by Gassend et al. [23] as a histogram. Holcomb et al. [78] used the metric to investigate the effect of aging and temperature by plotting the value of this metric against the value at reference environmental conditions.

Another way to investigate the intra-class distribution is to count the number of bit positions or challenges that did not reach a predefined level of stability m_l:

$$n_{c,d,e} = \sum_{b=1}^{B} \begin{cases} 1 & \text{if } m_l \leq m_{b,c,d,e} \leq (1 - m_l) \\ 0 & \text{otherwise.} \end{cases} \tag{2.44}$$

In comparison to the previous approach, this yields a more condensed report, because a single count replaces B stability estimates. For example, Gassend et al. [23] reported the ratio of challenges that were read incorrect at least once, but did not state whether this included temperature variation or multiple devices. Note that the metric "number of unstable bits" [6] used by Su et al. sounds like it would also refer to the number of bit positions that do not reach a predefined stability level, but has in fact been defined there as the mean intra-class HD. In contrast, the "distinct unstable bits" reported in a histogram by Maiti et al. [70] refer to the ratio of bit positions that were read incorrect at least once similar to the notion of Gassend et al. above. In [57], Maiti et al. use both "number of unstable bits" and "distinct unstable bits", where the former is explained to refer to the mean intra-class HD as used by Su et al. [6], while the latter can only be assumed to follow the definition in [70] based on the reported values. Kömürcü and Dündar

[81] furthermore suggested to use “stable bit count”, defined as the number of bit positions that were read correct in every access, as a metric.

Holcomb et al. [78] investigated the effect of aging on an SRAM PUF using the HW along bit positions, where different amounts of aging are represented in this work’s notation as different environmental conditions:

$$m_{d,e} = \frac{1}{B} \sum_{b=1}^{B} x_{b,d,e} \tag{2.45}$$

The values were then compared to the HW at reference environmental conditions and reported in a table. As the approach is targeted at single-challenge PUFs, it did not consider challenges. Holcomb et al. also did not state whether only a single access was made under each environmental condition or multiple accesses were processed into the single value. Böhm and Hofer [79] performed a similar experiment, where the effect of remanence rather than aging was investigated by comparison of the HW at reference conditions to the HW after writing either all cells to 0 or all cells to 1 and performing a one second power removal.

2.5. Probabilities And Correlations

2.5.1. Probability of (In-)Correct Identification

A probability metric that incorporates both intra-class and inter-class performance is the probability of misidentification respectively probability of correct identification. Lofstrom et al. [5] already discussed this issue and presented a figure about the number of distinguishable devices over response length and expected percentage of drifting response bits if “the application can tolerate a 10‰ chance of a single false-positive or false-negative [. . .] in the entire database” [5], but did not provide a formula or sufficient explanation to reproduce how the numbers are to be calculated. Suh and Devadas [32] provided false acceptance rate and false rejection rate, together with the probability that too many bit errors occur and an incorrect key is produced. They did not provide formulae, but explained that their values are based on the binomial distributions they fitted to their intra-class and inter-class HD distributions.

Binomial / Binomial Approaches

Lee et al. [12] calculated the probability of correct identification from two binomial distributions with interchip variation $\overset{\text{Lv}}{p}$ respectively noise $\overset{\text{Ln}}{p}$ as success probabilities, see Sec. 2.2.2 for their definition. They provide

$$\overset{\text{Lc}}{p} = \left(\overset{\text{Ld}}{p}(t, C, \overset{\text{Lv}}{p})\right)^{\binom{D}{2}} \left(\overset{\text{Lb}}{p}(t, C, \overset{\text{Ln}}{p})\right)^{D} \tag{2.46}$$

to calculate the probability that “any two chips do not agree on at least one (equals $(2t + 1)$ minus two time t) measured response and can be identified from one another” [12]. The two underlying binomial distributions are: First, the probability that any two devices differ in their reference response to at least $2t + 1$ challenges

$$\overset{\text{Ld}}{p}(t, C, \overset{\text{Lv}}{p}) = 1 - \sum_{i=0}^{2t} \binom{C}{i} \left(\overset{\text{Lv}}{p}\right)^{i} \left(1 - \overset{\text{Lv}}{p}\right)^{C-i} . \tag{2.47}$$

Second, the probability that a device returns an incorrect response for at most t out of C challenges during an authentication procedure

$$\overset{\text{Lb}}{p}(t, C, \overset{\text{Ln}}{p}) = 1 - \sum_{i=t+1}^{C} \binom{C}{i} \left(\overset{\text{Ln}}{p}\right)^{i} \left(1 - \overset{\text{Ln}}{p}\right)^{C-i} . \tag{2.48}$$

Lim et al. [31] distinguished between the authentication and identification use case in a similar discussion. If the PUF is used to authenticate a given device, the probability that the device is not recognized due to noise is claimed to be

$$\overset{\text{Lr}}{p}(t, C, \overset{\text{Ln}}{p}) = \sum_{i=t+1}^{C} \binom{C}{i} \left(\overset{\text{Ln}}{p}\right)^{i} \left(1 - \overset{\text{Ln}}{p}\right)^{C-i} , \tag{2.49}$$

so the probability to observe more than t errors in a binomial distribution with $\overset{\text{Ln}}{p}$ as error probability. The probability that another device can impersonate it, however, is given as

$$\overset{\text{La}}{p}(t, C, \overset{\text{Lv}}{p}) = \sum_{i=0}^{t} \binom{C}{i} \left(\overset{\text{Lv}}{p}\right)^{i} \left(1 - \overset{\text{Lv}}{p}\right)^{C-i} , \tag{2.50}$$

which is the probability to observe at most t different responses with $\overset{\text{Lv}}{p}$ as the probability that the response of another device differs. If the PUF response should identify one out of D devices in a database, the probability that the correct device is picked is given as

$$\overset{\text{Li}}{p}\left(C, \overset{\text{Ln}}{p}, \overset{\text{La}}{p}(t, C, \overset{\text{Lv}}{p})\right) = \sum_{i=0}^{C} \binom{C}{i} \left(\overset{\text{Ln}}{p}\right)^{i} \left(1 - \overset{\text{Ln}}{p}\right)^{C-i} \left(1 - \overset{\text{La}}{p}(i, C, \overset{\text{Lv}}{p})\right)^{D-1} . \tag{2.51}$$

Binomial / Uniform Approach

In contrast to the approaches by Lee et al. and Lim et al., the formula provided by Su et al. [67] for the probability of misidentification contains the assumption that true responses are uniformly distributed in $\mathcal{X}$, which only holds for an ideal PUF. Their version,

$$\overset{\text{Sm}}{p}\left(\overset{\text{Su}}{p}\right) = \sum_{i=0}^{B}\left[\binom{B}{i}0.5^{i}(1-0.5)^{B-i}\sum_{i/2}^{i}\binom{i}{i/2}\overset{\text{Su}}{p}^{i/2}\left(1-\overset{\text{Su}}{p}\right)^{i-i/2}\right],$$

contains therefore only one probability parameter, $\overset{\text{Su}}{p}$, which is defined as "the fraction of unstable bits" [67]. The formula is obviously inconsistent since the second sum does not contain a running index. However, given the explanation in [67] that the inner sum calculates the probability that a device produces a response further than $\lceil \mathrm{i}/2 \rceil$ bit away from its true response, which is then weighted with the probability that the HD to the next device is i, a reasonable correction according to [10] is

$$\overset{\text{Sm}}{p}\left(\overset{\text{Su}}{p}\right) \leq \sum_{i=0}^{B}\left[\binom{B}{i}0.5^{B}\sum_{j=\lceil i/2\rceil}^{B}\binom{B}{j}\overset{\text{Su}}{p}^{j}\left(1-\overset{\text{Su}}{p}\right)^{B-j}\right]. \qquad (2.52)$$

Note that although Su et al. present a line plot of the probability of misidentification dependent on the number of produced devices, above formula produces the probability that a randomly selected device is incorrectly identified as another one without consideration of how many devices have been produced. This behavior is identical to the formulae by Lee et al. and Lim et al. No explanation is given by Su et al. how the plot that considers the number of devices was produced.

2.5.2. Error Rate of Best Imitation

To measure the unpredictability of a PUF candidate, the error rate of the best available imitation can be used, i.e. the probability that the imitation produces an incorrect guess. This probability is typically compared to the intra-class distribution to evaluate whether the errors from the imitation could go unnoticed as readout errors of the actual PUF instance. Comparison to the inter-class distribution shows how well the imitation matches the device to be imitated in comparison to other devices of the same PUF design. Examples can be found in Gassend et al. [23] and Lee et al. [12]. Kalyanaraman et al. [27] reported the metric in comparison to an arbiter PUF to support their argument of less vulnerability to modeling attacks.

2.5.3. Bit Error Rate

To provide a single bit error rate as a metric, a common assumption is that bit errors, i.e. response bits that in this access do not match their corresponding bit of true response, occur independently of each other and the probability of such a bit error is equal among bit positions, challenges, and devices. The bit error rate is then typically inferred from the intra-class HD distribution, e.g. from its maximum by Guajardo et al. [17] or its mean by Maiti et al. [57], Maes et al. [34].

Maes [83] provides a more sophisticated approach than an average bit error rate based on the observation that error probability differs strongly between response bit positions. Instead of an average error rate, the approach estimates a PMF of the number of bit errors within a certain block of response bit positions.

2.5.4. Correlation of Adjacent Bit Positions

To investigate whether adjacent PUF cells are dependent, one may calculate their correlation or the conditional probability. An example was given by Su et al. [67], where for both ASIC layouts the conditional probabilities for a 0 and for a 1 at the upper, lower, left, or right position, given that the current position is 0 or is 1, averaged over all positions, was reported.

2.5.5. Autocorrelation Along Bit Positions

Another metric to investigate whether the response bits of a device are dependent is the autocorrelation function along bit positions as presented by Böhm and Hofer [79]. Their formula holds under the assumption that the response bits of a device originate from a single stochastic process that is at least wide-sense stationary, so mean and variance are constant among all bit positions. Instead of a data normalization, i.e. a subtraction of the mean and division by the variance of the sample, they mapped response bits from $\{0, 1\} \rightarrow \{-1, 1\}$ in an attempt to obtain a zero mean, unity variance sequence. Although the formula given in [79] did not divide by the number of multiplications, the textual description and the presented results suggest this was by accident, so it is reported here including the division. In this work's notation and under the assumption that it is supposed to be applied on the true response although this is not stated in [79], the metric can be written as

$$\overset{\text{Bc}}{r}_{d}(i) = \frac{1}{B - |i|} \sum_{b=\left\{\substack{1 \text{ if } i \geq 0 \\ 1-i \text{ if } i<0}\right.}^{\substack{B-i \text{ if } i \geq 0 \\ B \text{ if } i<0}} (2x_{b,d} - 1)(2x_{b+i,d} - 1), \tag{2.53}$$

where $|i|$ denotes the absolute value of i, which is the lag, i.e. the amount of shift introduced when the response is compared to itself. As it has been introduced for single-challenge PUFs, it does not consider challenges. If the response bit positions are chosen independently at random, $\overset{\mathrm{Bc}}{r}_d(i)$ should look like a noise floor in a plot. A peak for some lag i would indicate that there exists some pattern of length i among the response bit positions. If i matches for example the length of a row in a matrix like layout of PUF cells on the die, a peak at i or $-i$ would indicate that every row produces a similar pattern of 0s and 1s.

2.6. Information Theoretic Quantities

2.6.1. Entropy of Inter-Class Distribution

To measure the unpredictability of a PUF, entropy is an intuitive choice. However, multiple ways to calculate different notions of entropy from the same data can be found in the literature. While Pappu [2] already discussed *bitwise* entropy and stated that a Gabor hash maximizes entropy because the probabilities of individual response bit positions to be set is close to 0.5, bitwise entropy as a metric was explicitly introduced by Maes [82], where it corresponded to one of multiple attacker models.

The entropy metrics by Maes were based on identifiers, which may combine the responses to multiple challenges. Therefore, index l is used to iterate over the L bits in the identifier to correctly represent the metrics. With $L = BC$ in [82], the metrics automatically boil down to operate either on plain response bit positions $b \in \{1, \dots, B\}$ for single-challenge PUFs or plain challenges $c \in \{1, \dots, C\}$ for multi-challenge PUFs with single-bit output.

According to Maes [82], the identifiers bear some unknown amount of entropy, for which only an upper bound can be stated. This upper bound depends on the amount of knowledge an attacker might gain about the PUF design. Due to the concept of PUFs, an attacker that possesses all design files, according to Kerckhoff's principle, but nothing else must assume that the responses are perfectly uniformly distributed – as long as there is no systematic design flaw – and thus can only use the rather trivial bound

$$\overset{\mathrm{Mi}}{H} = L = BC. \tag{2.54}$$

Such an attacker was called "completely ignorant" [82] of the actual distribution of the PUF responses. The metric therefore contains superset Mi as abbreviation for *Maes, ignorant*. Once an attacker has access to responses, e.g. from other devices built according to the same design, increasingly sophisticated conclusions can be drawn that may increase her ability to guess the response of the attacked device correctly. As a first step, a "global bias entropy bound" [82], labelled Mg for *Maes, global*, was defined as

$$\overset{\text{Mg}}{H} = L\,\mathrm{H}\left(\mathfrak{B}(1, m)\right) \tag{2.55}$$

with m the global dataset mean from (2.42) and $\mathrm{H}\left(\mathfrak{B}(1, m)\right)$ the Shannon entropy[2] of a Bernoulli distribution with success parameter m. In a next step, the bias of individual, but sill independent, response bit positions and challenges was considered, which provided the bound

$$\overset{\text{Mb}}{H} = \sum_{l=1}^{L} \mathrm{H}\left(\mathfrak{B}(1, m_l)\right) \tag{2.56}$$

with m_l the mean along devices according to (2.32) and appropriate mapping from l to b and c. The superset Mb abbreviates *Maes, bitwise*. As a third option, the joint distribution of pairs of bits in the identifier was taken into account, which yields

$$\overset{\text{Mj}}{H} = \sum_{l=1}^{L} \mathrm{H}\left(\mathfrak{B}(1, m_l)\right) - \sum_{l=1}^{L-1} \mathrm{I}\left(X_l, X_{l+1}\right), \tag{2.57}$$

where $\mathrm{I}(\cdot, \cdot)$ denotes the mutual information between two random variables and X_l is the random variable that produces the bits at position l in the identifiers of all devices. The superset Mj abbreviates *Maes, joint*. Finally, Maes described an attacker with an appropriately trained model that provides estimations for each bit in the identifier, which are correct with probability $\overset{\boxplus}{p}$. This led to the "model entropy bound" [82]

$$\overset{\text{Mp}}{H} = L\,\mathrm{H}\left(\mathfrak{B}(1, \overset{\boxplus}{p})\right) \tag{2.58}$$

with superset Mp as abbreviation for *Maes, probability*. Note that although the formula is very similar to (2.55), the success parameter bears different meaning.

2.6.2. Min-Entropy of Intra-Class Distribution

Holcomb et al. [78] used the min-entropy of the intra-class behavior to estimate how much entropy a TRNG built from the noise during readout could deliver. To obtain acceptable probability estimates for 4 096 bit responses from just 100 accesses, they split the responses into bytes and modeled each byte as an independent RV.

[2]Note that it is not perfectly clear which entropy measure, e.g. Shannon entropy or min-entropy, was used for this metric in [82]. However, Shannon entropy may be concluded from the fact that it uses H, while later chapters in [82] use $\tilde{H}$ for min-entropy.

This means X was substituted by the random vector $\underline{Z} = \left(\, Z_1 \; \cdots \; Z_{512} \, \right)$, where RV Z_i covered the i^{th} response byte, so response bit positions $b = \{8i-7, \ldots, 8i\}$, as a discrete 256 outcome distribution. For brevity and since the outcomes of a Z_i can be perceived as atomic symbols here, they are denoted as integers rather than bit strings, i.e. $\mathscr{Z} = \{0, 1, \ldots, 255\}$. The min-entropy of the intra-class distribution according to Holcomb et al. may then be written as

$$\overset{\text{Ha}}{H} = -\log_2 \left(\prod_{i=1}^{512} \max_{z \in \mathscr{Z}} \left(\mathrm{P}(Z_i = z) \right) \right) , \tag{2.59}$$

where the probabilities are presumably estimated from the sample counts, i.e. how often byte i turned out as 0, 1, 2, etc. on a device within the dataset. Although targeted at the secondary use for a TRNG, it indirectly is a metric for the PUF's quality, because it describes the amount of noise during readout. For a pure PUF use case, this metric should therefore be as close to zero as possible.

2.6.3. Randomness and Steadiness

In addition to the FHD based metrics listed in Sec. 2.3, Hori et al. [72] provided two more metrics based on min-entropy. Randomness has been defined as

$$\begin{aligned} \overset{\text{Hr}}{H}_d &= -\log_2 \max({m_d}', 1 - {m_d}') \text{ , where} \\ {m_d}' &= \frac{1}{AKL} \sum_{l=1}^{L} \sum_{k=1}^{K} \sum_{a=1}^{A} x_{a,l,k,d} \end{aligned} \tag{2.60}$$

and measures the overall amount of min-entropy in the dataset. Note that although it was defined using identifiers, the result is unaffected by a substitution of $b \geq 1$ and c for l and k, because the mean operations are commutative. This allows to operate directly on responses without construction of identifiers.

Steadiness has been defined as

$$\overset{\text{Hs}}{H}_d = 1 + \frac{1}{KL} \sum_{k=1}^{K} \sum_{l=1}^{L} \log_2 \max(m_{l,k,d}, 1 - m_{l,k,d}), \tag{2.61}$$

where for the same reason as above $m_{l,k,d}$ can be replaced by $m_{b,c,d}$ from (2.44) with e omitted. It is supposed to measure how stably a PUF cell produces its response, which means the PUF cell should ideally return the same response in all accesses. Since this would cause a min-entropy of zero, but Hori et al. designed their metrics to produce zero for the worst performance and unity for an ideal PUF, the metric subtracts the min-entropy from unity.

CIs for both metrics were provided by application of the CLT. For Randomness, Hori et al. applied the CLT on $m_d{}'$ to avoid the nonlinearity of the logarithm. Then $m_d{}'$ follows a normal distribution, which allows to calculate a CI for the population mean from the percentiles of the t-distribution and then apply the max-operator and logarithm to obtain CI limits for the min-entropy. For Steadiness, the CLT was instead applied on $\overset{\text{Hs}}{H}_d$ directly, and a CI was calculated without intermediate step.

2.6.4. Mutual Information

Ignatenko et al. [77] proposed to use the maximum secrecy rate as a metric, which equals the mutual information between the true response and another measurement. Focused on optical PUFs, they first proved how to estimate the limit entropy for two dimensional speckle patterns using the context-tree weighting (CTW) algorithm to then be able to calculate the mutual information from the individual limit entropies of true response and measurement and the joint limit entropy of both. The mutual information was reported in [77] for individual devices and on average. Mutual information has since been used without discussion of limits and two dimensional structures. Guajardo et al. [17] reported an average mutual information of 0.76, although it is not clear whether the average is applied only among devices or also among accesses since mutual information takes only two bitstreams at a time similar to a HD operation. Schrijen et al. [76] explained that to calculate the mutual information using CTW, they compressed the concatenation of true responses from all devices, while giving the concatenation of responses from all devices from another access as context to the CTW algorithm. With the second access performed under various environmental conditions, the minimum, average, and maximum mutual information was given in a table.

2.6.5. Compressability of Responses

Apart from estimation of the mutual information, compression with CTW can also be used to estimate the entropy within a device's response. As an example, Katzenbeisser et al. [15] stated to individually compress one file per device and environmental condition, where the file contains a concatenation of responses for all challenges. Although this should provide as many numbers as analyzed devices, only one number was reported for each environmental condition and PUF type and no explanation was made whether this refers to the mean, maximum, or another measure of distribution. A similar issue exists in Schrijen et al. [76], who described to compress the true response of each device individually, but did not explain how these values per device are processed into the single value that was tabulated.

2.6.6. Conditional Entropy Between Bit Positions On Same Device

Katzenbeisser et al. [15] used conditional entropy and conditional min-entropy to estimate the remaining unpredictability given partial knowledge of the response. Partial knowledge in this case means that an attacker knows the response for – depending on the PUF type – similar challenges or adjacent cells on the same device. The resulting values were reported in histograms over devices for mutliple environmental conditions individually.

2.6.7. Identifying Information

Holcomb et al. [78] suggested the amount of identifying information as a metric. It was deduced from the response length and the number of devices for which successful identification had been shown. Thus, if for all D devices and in every access the obtained response was closer to the true response of the accessed device than any other true response,

$$\overset{\mathrm{Hi}}{H} = \frac{\log_2(D)}{B} \tag{2.62}$$

was said to be a lower bound on the amount of identifying information per cell that this PUF candidate provides. Holcomb et al. then related the above metric to the amount of die area consumed to reflect that a PUF candidate may offer less identifying information per cell than some other candidate, but may still be more economic if it is much smaller than the competitor. Precisely, the suggested metric is the cell area divided by the amount of identifying information as calculated above.

2.7. Distribution of Analog Response Values

For some types of PUF such as the RO PUF or coating PUF, analog response values are available. Analysis of these values usually provides more information than an analysis of the binary response strings. It also enables to assess the PUF candidate before a bit extraction algorithm such as pairwise comparison is applied. However, it is limited to the few types of PUF that provide analog data and therefore less common than metrics that can be applied to binary responses. It is comprehensively performed in e.g. [70, 80]. Gassend et al. [7] analyze the frequency of their self-oscillating loop circuit among temperature and compare the associated measurement noise with the difference between multiple devices.

Maiti et al. [70] use the analog domain to define the true frequency of an RO as the mean among all accesses to it as

$$\bar{\xi}_{b,d} = \frac{1}{A} \sum_{a=1}^{A} \xi_{a,b,d}. \tag{2.63}$$

The binary true response is then found by application of the bit extraction algorithm on these true frequencies. They furthermore calculated the noise level of a device as the mean among ROs of each RO's standard deviation in frequency among accesses

$$s_d = \frac{1}{B} \sum_{b=1}^{B} \sqrt{\frac{1}{A-1} \sum_{a=1}^{A} \left(\xi_{a,b,d} - \bar{\xi}_{b,d}\right)^2}. \tag{2.64}$$

and reported the noise levels in a histogram. They also provided histograms of the mean and standard deviation of true frequency among all ROs on a device, and how the mean frequency of a device is distributed among devices.

Heat maps of RO's frequencies can be found for example in [33, 57].

2.8. Post-Processing Aware Metrics

The previously described metrics mostly operate on the PUF responses before any post-processing is applied and typically measure intra-class and inter-class behavior separately. A reasonable alternative, however, is to evaluate the unpredictability after appropriate post-processing is applied and thus evaluate both aspects together. Which post-processing is appropriate is determined by the reliability of the PUF candidate design and the reliability goals of the intended application. This approach is therefore more suited to decide whether a given PUF candidate design meets the demands of the intended application, e.g. some secure key storage application as described in Sec. 1.3.3. It is less helpful to identify a particular flaw in a PUF candidate's design.

Common post-processing approaches for key storage applications are *fuzzy extractor* and *fuzzy commitment*. Both use an ECC, where the input to the ECC is a full entropy random binary vector $\underline{u}$. The output codeword $\underline{v}$ is then XORed onto a binary string $\underline{x}$, produced by the PUF, to obtain the helper data $\underline{y}$. Since the helper data is public information, the entropy that remains for $\underline{u}$ and $\underline{x}$ even if an attacker knows $\underline{y}$ is the figure of merit measured by the following metrics. It is affected by the unpredictability of the PUF responses as well as the PUF response's reliability, since less reliability requires a stronger ECC with shorter input, which means less entropy from $\underline{u}$.

2.8.1. (n-k) Bound

The so-called (n-k) bound has been provided by Dodis et al. in their original publication on fuzzy extractors [53] and holds equally for other HDSs such as the fuzzy commitment scheme. It is a general construction not dedicated to PUFs but any noisy data source, with biometrics given as example in [53]. Therefore it does not deal with details of the source such as whether $\underline{x}$ is an identifier comprised of multiple responses from a multi-challenge PUF or just the response of a single-challenge PUF. Instead it assumes some appropriate entropy estimation of the data source is feasible. Although it only provides a rough upper bound on the amount of entropy lost in the HDS, it is used frequently in the PUF field due to its simplicity.

The metric resembles a simple balance sheet: On the ingoing side, there is the entropy provided by the noisy source, e.g. the PUF, commonly denoted m, and the entropy of the random binary vector $\underline{u}$, commonly denoted k.[3] The outgoing side contains: The entropy revealed to an attacker via the helper data $\underline{y}$, commonly denoted n; for the case of fuzzy extraction instead of fuzzy commitment the entropy loss o in the hash function; and the entropy l that remains about $\underline{u}$ for an attacker with access to the helper data $\underline{y}$. The balance hence is

$$m + k = n + o + l \tag{2.65}$$

and the overall entropy loss is

$$m - l = n - k + o. \tag{2.66}$$

Neglecting the entropy loss of the hash function, o, the overall entropy loss of the construction is $n - k$, which explains the name of the metric.

As mentioned above, this balance sheet only provides a rough upper bound on the entropy loss, which holds with equality only if the binary string $\underline{x}$ produced by the PUF has full entropy, i.e. $m = n$ [36, 53]. An intuitive affirmation for this can be found in the following trivial example: Assume a 3-repetition code which has according to the (n-k) bound an entropy loss of $3\,\text{bit} - 1\,\text{bit} = 2\,\text{bit}$. The reason is that by definition of a repetition code $\underline{v} \in \{000, 111\}$, so an attacker may conclude from e.g. $\underline{y} = 010$ that $\underline{x} \in \{010, 101\}$. Since there are only two

[3]Caveat: k, l, m, n, o may not be mixed up with their meaning in other sections or chapters of this work. They are used here mainly to explain why the metric is called the (n-k) bound. ECCs are typically described by a tripel (n, k, t) where n is the code word length, k the message length, and t the number of arbitrary bit errors that can be corrected. More than t bit errors may be corrected if decently located and the ECC is not a maximum distance separable (MDS) code.

options left, and if they are equiprobable, the remaining entropy is 1 bit.[4] Thus the entropy of the second and third bit position in $\underline{x}$ are lost and only that of the first bit position remains. Let the PUF now contain a flaw that makes the third bit in any $\underline{x}$ equal the first. Such $\underline{x}$ obviously contain only 2 bit of entropy, because there are only four equiprobable outcomes with non-zero probability: 000, 010, 101, and 111. Since $m = 2$ bit now, the (n-k) bound suggests all entropy is lost, i.e. knowing $\underline{y}$ implies knowing $\underline{x}$. An attacker, though, would still face 1 bit of entropy: While the flaw limits the number of possible $\underline{y}$, for any $\underline{y}$ there are still two options for $\underline{x}$, either $\underline{x} = \underline{y}$ or $\underline{x} = \neg\underline{y}$, where $\neg$ indicates one's complement a.k.a. bitwise inversion. From the other point of view, the dependency in an $\underline{x}$ due to the PUF's flaw overlaps with the dependencies introduced by the ECC, so there are two independent ways to infer the third bit of an $\underline{x}$. Thus the only additional insight an attacker gains from knowing $\underline{y}$ is the value of the second bit of $\underline{x}$, which limits the entropy loss to 1 bit, so the remaining entropy is again 1 bit.

2.8.2. Expected Conditional Min-Entropy

Exact Calculation

Dodis et al. [53] also presented a direct mathematical expression to measure the expected remaining min-entropy regarding $\underline{x}$ given some $\underline{y}$, which is somewhat more complex, though:

$$\overline{\mathrm{H}}_\infty(X|Y) = -\log_2\left(\mathop{\mathrm{E}}_{\underline{y}\leftarrow Y}\left[2^{-\mathrm{H}_\infty(X|Y=\underline{y})}\right]\right) \tag{2.67}$$

Note that, since Dodis et al. dealt with an abstract noisy source, the binary output string $\underline{x}$ is considered a single element of the set of outcomes $\mathcal{X}$ of RV X. X follows a discrete distribution with 2^L outcomes. This means that if the identifier passed on to the HDS by the PUF has e.g. four bits, $L = n = 4$ and $\mathcal{X} = \{0000, 0001, \ldots, 1111\}$ regardless of how many challenges are required to produce them. The innermost exponent, $\mathrm{H}_\infty\left(X|Y = \underline{y}\right)$, is the min-entropy conditional on a specific helper data word. To obtain the *expected* conditional min-entropy $\overline{\mathrm{H}}_\infty(X|Y)$, the former is enclosed with the expectation among helper data words. Although it might seem unintuitive to transfer the entropy into a probability before the expectation is performed and reverse this operation afterwards, this is necessary to avoid an overestimation of remaining conditional min-entropy by several orders of magnitude, cf. [84] for a detailed explanation.

[4] An alternative point of view is that, while knowing $\underline{y}$ does not enable an attacker to make a better guess for the first bit of $\underline{x}$, the attacker knows from the fact that all bits in $\underline{v}$ are equal that the second and third bit of $\underline{x}$ are either equal to or the inverse of the respective bits of $\underline{y}$ depending on whether the first bit is equal or the inverse.

Two simplifications of (2.67) have been given by Delvaux et al. [36]. They utilize the fact that the input to the ECC, a random number $\underline{u} \in \mathcal{U}$, $\underline{u} \leftarrow U$, is uniformly distributed, thus all code words $\underline{v} \in \mathcal{V}$ are equally probable, and combine this with Bayes' rule. For a general ECC,

$$\overline{\mathrm{H}}_{\infty}(X|Y) = -\log_2 \left(\frac{1}{|\mathcal{U}|} \sum_{\underline{y} \in \mathcal{Y}} \max_{\underline{v} \in \mathcal{V}} \mathrm{P}\left(X = \underline{y} \oplus \underline{v} \right) \right) . \tag{2.68}$$

For linear ECCs, the former can be further simplified to

$$\overline{\mathrm{H}}_{\infty}(X|Y) = -\log_2 \left(\sum_{\underline{\epsilon} \in \mathcal{E}} \max_{\underline{v} \in \mathcal{V}} \mathrm{P}\left(X = \underline{\epsilon} \oplus \underline{v} \right) \right) , \tag{2.69}$$

where $\underline{\epsilon}$ is a coset leader, i.e. an element of the minimum HW error vector space $\mathcal{E}$ so that $\{\underline{v} \oplus \underline{\epsilon} | \underline{v} \in \mathcal{V}, \underline{\epsilon} \in \mathcal{E}\} = \mathcal{Y}$. Using again a 3-repetition code as an example, code word space $\mathcal{V} = \{000, 111\}$ and error vector space $\mathcal{E} = \{000, 001, 010, 100\}$. Despite the simplification with regard to (2.67), calculation according to (2.68) still requires $2^n|\mathcal{U}|$ operations and according to (2.69) 2^n operations. Their direct practical use is therefore limited to ECCs with code word length n up to around 60 bit with today's computing resources.

If the HDS is built upon block ECCs in a way that splits PUF response and helper data into multiple blocks, the blocks are independent of each other and the expected conditional min-entropy may be calculated for each block individually and summed together. Given that the block size is smaller than the above mentioned limit on code word length, this allows to calculate the expected conditional min-entropy for realistic sizes of $\underline{x}$ and $\underline{y}$ already. However, ECCs with longer code words are preferred in practice because of their higher error correction capability.

Feasible Lower Bound for ECCs With Long Code Words and IID Source

To apply (2.68) or (2.69) to practically relevant ECCs with long code words, Delvaux et al. [36] provided an efficient approach to obtain a tight bound on them[5]. The key is an IID assumption about the source, i.e. the assumption that all bits in $\underline{x}$ originate from independent drawings of a $\mathfrak{B}(1, p)$ distribution. While [36] makes no statement on how to obtain p, other work that makes this assumption uses m from (2.42), e.g. [82]. The benefit is that the probability of occurence for any response word $\underline{x} \in \mathcal{X}$ now directly depends on the HW of $\underline{x}$, i.e. on the number of bits that turned out as 1, but not *which* response bit positions did so. Thus, $\mathcal{X}$ can

[5]A previous version of the summary given in this subsection appeared in [**85**].

easily be partitioned into $J = n + 1$ subsets φ_j with $j \in \{0, \dots, n\}$, where a φ_j contains all $\underline{x}$ with HW equal to j. Each subset φ_j then contains

$$|\varphi_j| = \binom{n}{j} \tag{2.70}$$

elements of $\mathcal{X}$ with equal probability of occurence

$$\overset{\varphi}{p}_j = \left(\max\left(p, \overline{p}\right)\right)^{B-j} \left(\min\left(p, \overline{p}\right)\right)^{j}, \tag{2.71}$$

where $\overline{p} = 1 - p$ and

$$\overset{\varphi}{p}_j > \overset{\varphi}{p}_{j+1}. \tag{2.72}$$

The subsets φ_j constitute a partition of $\mathcal{X}$ in the mathematical sense since each response $\underline{x}$ belongs to exactly one subset, thus the subsets φ_j are mutually exclusive and collectively exhaustive. The subsets φ_j therefore constitute a concise and practically usable representation of the probability distribution of X.

This is useful because for every given helper data word $\underline{y}$ respectively coset leader $\underline{\epsilon}$, the max-operator in (2.68) respectively (2.69) selects the most probable element in the response space $\mathcal{X}$ that is reachable by addition of a code word $\underline{v} \in \mathcal{V}$. Since the input to the ECC $\underline{u} \in \mathcal{U}$ is uniformly distributed, and thus all code words $\underline{v}$ are equally probable, this leads to the same $\underline{x} \in \mathcal{X}$ being selected for every $|\mathcal{U}| = |\mathcal{V}|$ elements in helper data space $\mathcal{Y}$. This is visualized by the columns in Fig. 2.1. For example, $\underline{x} = 100$ is the most likely response $\underline{x}$ within reach for both $\underline{y} = 100$ and $\underline{y} = 011$; via $\underline{v} = 000$ and $\underline{v} = 111$, respectively. It is thus sufficient to consider $^{2^n}/_{|\mathcal{U}|}$ elements of $\mathcal{X}$ to cover the entire helper data space $\mathcal{Y}$, visualized by the two halves in Fig. 2.1, to calculate either of (2.68) or (2.69).

Which $^{2^n}/_{|\mathcal{U}|}$ elements of the response space $\mathcal{X}$ to consider depends on the specific ECC. For repetition codes with odd code word length n, which are able to correct up to $t = \lfloor \frac{n}{2} \rfloor$ bit errors, $\bigcup_{j=0}^{t} \varphi_j$, i.e. the $^{2^n}/_{|\mathcal{U}|}$ *most probable* responses $\underline{x}$, are the correct choice, which corresponds to the left half in Fig. 2.1. The reason is that the IID assumption means

$$\mathcal{E} = \begin{cases} \bigcup_{j=0}^{t} \varphi_j & p < 0.5 \\ \{\underline{x} \oplus 111 | \underline{x} \in \bigcup_{j=0}^{t} \varphi_j\} & p > 0.5 \end{cases}. \tag{2.73}$$

For other codes, however, such as a (15,5,3) BCH-code, in addtion to $\bigcup_{j=0}^{3} \varphi_j$, 420 $\underline{x}$ from φ_4 and 28 from φ_5 are required. To choose 448 $\underline{x}$ from φ_4 instead, thus again the $^{2^n}/_{|\mathcal{U}|}$ *most probable* $\underline{x}$, overestimates the probability for 28 out

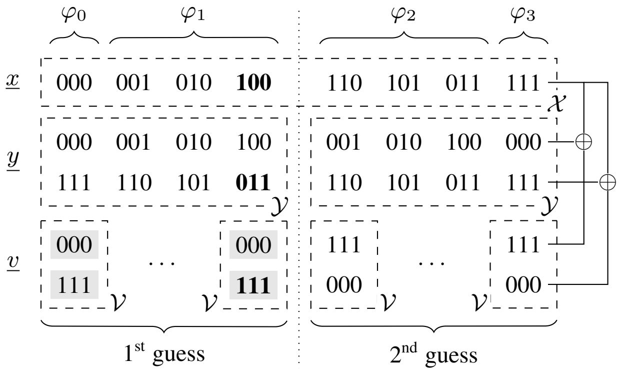


Figure 2.1.: Visualization of the relationship between PUF output, helper data, and codeword of typical HDSs for key storage using a 3-repetition code and $p < 0.5$; reproduced from [**85**]. The most probable $\underline{x}$ – which is the one selected by the max-operator in (2.68) and (2.69) and would be the best guess for an attacker that knows $\underline{y}$ and the distribution of X – for every possible $\underline{y}$ can be found in the left half of the figure. With $\underline{y} = 011$, for example, the best guess is $\underline{x} = 100$ thus $\underline{v} = 111$, $\underline{u} = 1$.

of 1024 $\underline{x}$. In general, *always* choosing the $^{2^n}/_{|\mathcal{U}|}$ most likely responses $\underline{x}$ thus provides an upper bound for the sum in (2.68), (2.69), which leads to a lower bound for the expected conditional min-entropy $\overline{\mathrm{H}}_\infty(X|Y)$. The bound holds with equality for MDS ECCs.

Since $^{2^n}/_{|\mathcal{U}|}$ operations may still be infeasible, it is necessary to utilize the fact that all responses $\underline{x}$ in a subset φ_j have the same probability of occurence. Therefore the contribution of a subset φ_j to the sum is equal to the product of its cardinality $|\varphi_j|$ and the probability $\overset{\varphi}{p}_j$. This effectively allows to process $|\varphi_j|$ elements of the response space $\mathcal{X}$ at once, so that for a linear (n, k, t) block code t or $t + 1$ operations suffice. For the above mentioned (15,5,3) BCH-code, this would mean a reduction in computational cost from 2^{15} down to 4. However, it only applies if an IID assumption on the binary string passed from the PUF to the HDS holds. A solution to overcome this limitation is provided in Sec. 4.4.

3. Evaluation of Existing Metrics for Reliability and Unpredictability

The previous chapter provided an overview on previously proposed or used metrics to test a PUF candidate's CRB. In this chapter, a comparison and evaluation of these metrics follows. Starting with a general discussion of issues in the field of PUF testing, it continues with an algebraic assessment of overlap, sensitivity, and blind spots among common metrics. The introduction of statistical models for PUF candidates then paves the way to evaluate the metrics from a statistical point of view, e.g. regarding expectation and variance for an ideal PUF, achievable CIs, etc. It also enables to use statistical hypothesis testing as known from test suites for RNGs, whose applicability to PUFs is evaluated at the end of this chapter.

3.1. Lack of a Standardized Assessment Procedure

One of the most obvious issues revealed by the previous chapter is the large variety of metrics. While a general trend towards metrics based on HD can be identified, there remains sufficient variation in the way HD is applied on the data to render the results unfit for fair comparison. Furthermore, even publications that try to establish a standard set of metrics, such as [70, 72], did not achieve sufficiently widespread adoption, which leaves the evaluation of a PUF candidate's performance a highly fragmented area of research, where many authors follow their own ideas instead of a common standard among researchers.

In this situation, confusion is further increased by subtle adjustments to or possibly incorrect application of previously published and named metrics without clear indication of the changes. An example is the inter-class HD distribution, which may be calculated on the basis of device pairs and separately for each challenge as in (2.2), or over devices *and* challenges as in (2.3), or with the dimension of challenges constituting a string as in (2.23), or on the basis of identifiers, cf. Sec. 2.3. In such cases the reader is left to make vague guesses on the details of calculation from things such as the granularity of plots, or the total of histogram bars.

There are also multiple examples where the name of a previously published metric is entirely redefined: The name Uniqueness has at least three different definitions, by Maiti et al. [70], by Hori et al. [72], and by Merli et al. [33]. Reliability is also defined at least twice by Maiti et al. [70] and Merli et al. [33]. The conflicts in definition also exist in the opposite way, though. For example, Su et al. refer to the previously mentioned metric of Reliability as "number of unstable bits" [6], which is not to be confused with what Maiti et al. refer to as "distinct unstable bits" [70], because despite their similar name they differ substantially in meaning and usually also in value for the same design. See Sec. 2.4.2 for definitions of these metrics.

Furthermore, the variations or newly invented metrics are more than once not properly defined by a mathematical formula, but by a vague description in words. One example is the use of the term "average", which in statistical language may refer to any kind of measure that reports a typical value of a distribution such as arithmetic mean, median, or even mode. Another example are the "noise" and "interchip variation" as used by Lee et al. [12] and Lim et al. [31], where only a short textual description is given that does not allow to write down a clear mathematical formula to calculate them. Where formulae are given, they are sometimes incomplete, e.g. the formula for probability of misidentification in [6], or that for autocorrelation in [79]. On the question of compression for entropy and mutual information estimation, Ignatenko et al. [77] prove that CTW achieves optimality for infinitely long sequences even if they are two dimensional, but omit an exact description of how to apply the available implementation on the data to calculate the mutual information instead of merely compressing a response. Katzenbeisser et al. [15] state that they compress files comprised of all responses of a device to all measured challenges, but report only a single value without explanation how the values for each device where processed into a single one. Guajardo et al. [17] report a mutual information without explaining their calculations. Kömürcü and Dündar [81] propose a metric based on the correlation between the bit position based inter-class HDs and a Gaussian distribution, but give no further explanation how to calculate this. An explanation would be necessary, though, since all contemplable statistical measures – such as covariance, correlation, or correlation coefficient – are defined for two sequences of data, not a sequence and a distribution. They may actually intend a test for normality such as the Anderson-Darling test, but this remains a guess since no further information is provided on the matter. Failure to explain how true responses are defined is an issue in e.g. [6, 32, 56, 79]–[81].

Finally, the lack of standardized or at least properly defined metrics as described above is in particular obstructive for PUF research because only few authors publish their raw data. Despite the fact that the publication of raw data is good scientific practice, it would allow to cross-check the claimed results. Published raw data would also enable to test novel metrics or new flavors of existing metrics with existing data as well as compare new PUF candidates to previously published ones by equally computed numbers, even if the previous publication did not disclose their metric definitions. Among the few notable exceptions are Maiti et al. [70], Hori et al. [73], Wilde [**74**], and Hesselbarth et al. [**86**], who provide their data online for download, as well as Su et al. [67] who include a table in their paper that lists all the true responses, though not the readings under different temperature or supply voltage.

3.2. Misconception Regarding Separation of Intra-Class and Inter-Class HD Distributions

The distributions of intra-class and inter-class HD may be displayed in a combined histogram, so a gap or overlap between both distributions is easier to notice. Sometimes, e.g. in [2, 78], this is accompanied by the claim that if the distributions do not overlap, all devices can be correctly identified. However, whether this claim is true depends on the way the inter-class HD distribution is built.

If the inter-class HD distribution is built as in (2.4), used for example by Holcomb et al. [66, 78], the claim indeed holds. In (2.4), the responses from all accesses to a device are compared with the true responses of all other devices, and this for every device in the dataset. Together with the intra-class distribution, the combined histogram then contains the HD between the response of every single access to every device in the dataset and every true response, whether it belongs to the same device, or to another. If the distributions then do not overlap, this means that for every access in the dataset to any device, the HD of the measured response to the true response of the correct device is smaller than to any other true response and hence the device can be correctly identified.

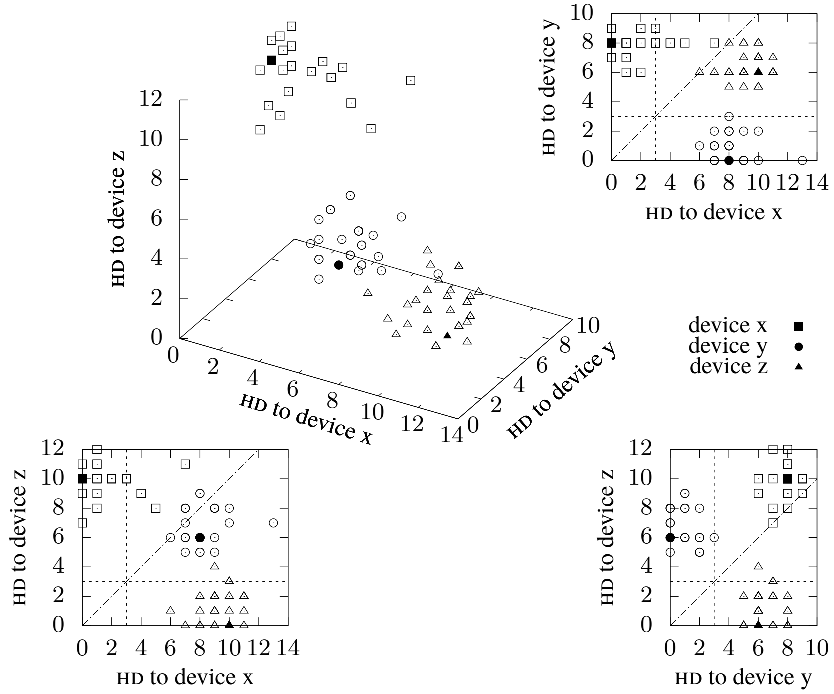
If the inter-class HD distribution is built upon true responses only, cf. (2.2) and e.g. [2], the claim of identifiability is not necessarily proven. If the distance between the true responses is e.g. in the range 0.4 to 0.6 and an access to a device may produce responses with e.g. at most 0.25 FHD, it might occur that the measured response is closer, e.g. just 0.15 FHD, to the true response of another device than its own. The largest intra-class HD would need to be less than *half* of the smallest inter-class HD to guarantee that no access in the dataset produces a response that might be incorrectly identified. So for this type of combined histogram it is insufficient that the distributions do not overlap. Instead, there must be a sufficiently wide gap to claim that all devices can be successfully identified for all accesses in the dataset.

Note that the inverse conclusion holds in neither case. All the devices may be correctly identifiable based on any single access in a dataset even if the distributions overlap. A crafted toy example that proofs this is shown in Fig. 3.1. It consists of three single-challenge devices that produce 16 bit responses, and 31 accesses are simulated for each device based on selected true responses and tuned noise shapes that represent the effects described below. Although the histograms suggest otherwise, the response from every access to any device in this dataset can be correctly identified based on minimal HD to its corresponding true response. This is most easy to verify by consulting the 2D projections, where a dash-dotted line is added to depict equal distance. Selecting for example device X, the square points lie above this line in both the XY and the XZ projections; this shows that despite the high intra-class FHD observed in some accesses, it is still less than the FHD to the other devices' true response. There are two reasons for that:

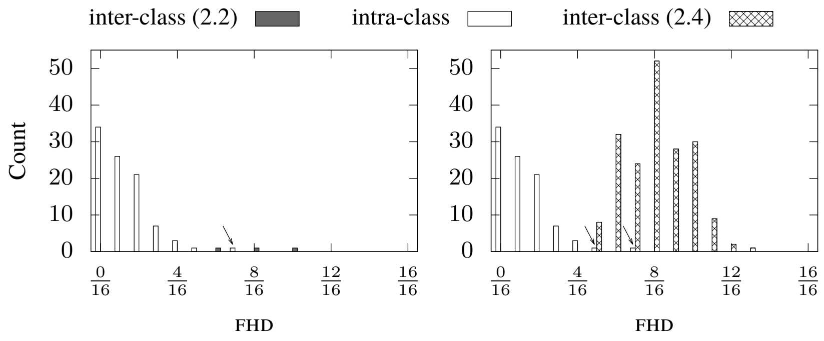
First, the histograms only visualize distance, but not direction. Devices Y and Z have an identical mean bit error rate of 7%, but it varies among response bit positions, i.e. there are response bit positions that are more stable and some are less stable. This is not uncommon for PUFs, because the error rate depends on how far e.g. the threshold voltages or RO frequencies were tossed apart by the manufacturing variations. This behavior even inspired lightweight error correction by identifying and masking out unreliable bits during enrolment[1]. Although difficult to imagine for more than three dimensions, each response bit position is an orthogonal dimension in the response space. Thus the cloud of points obtained by drawing a point in a B dimensional space for every access is not necessarily circular, but most likely oval except for the case where each response bit position has identical error rate. If, as it is the case in this crafted example, the oval shape happens to be oriented such that most noise occurs in response bit positions not essential to distinguish a pair of devices, they may be correctly identifiable although the HD due to noise exceeds even the HD of their true responses, which is 6 bit in this example. In the 3D plot in the middle of Fig. 3.1a, this is visible as the cloud of circular shaped points being located rather above than below the filled circle that represents the true response of device Y and the cloud of triangular shaped points being located rather behind the filled triangle than before.

Second, because the histograms are built over all devices, individual properties are no longer discernible. As error rates depend on the outcome of manufacturing variations, they may differ not only between response bit positions, but also between devices. Therefore some devices may show a wider intra-class HD distribution than others, but if those for the same reasons also tend to have a higher HD in true response to other devices, identification may still be possible because the measured responses are still closer to their own device's true response than any other. This is simulated by device X, which has an elevated bit error probability of 10% for all bit positions, but also a higher HD of its true response to those of others,

[1]This kind of error correction, named dark bit masking, was meanwhile shown to be insecure under certain circumstances, though, because it gives rise to helper data manipulation attacks.



(a) 3D plot and projections. A point is drawn for each access with a shape that corresponds to the device and at coordinates that represent the HD to all devices' true response. Filled points represent the true responses.



(b) Combined histograms. The overlap of the intra-class FHD distribution with either type of inter-class FHD distribution is highlighted by arrows.

Figure 3.1.: Crafted dataset where correct identification is possible for every access and every device despite overlap in HD distributions.

8 bit to device y and 10 bit to device z. An additional, related issue is that from the histogram alone one cannot tell whether a wide intra-class HD distribution stems from all devices producing highly incorrect responses from time to time, or just a few devices doing so quite often. The consequences are very different, though. In the former case, reliability improvement methods such as majority voting need to be implemented on all devices, which increases cost, identification duration, etc. In the latter case, highly unstable devices are easily identified in an end-of-line production test and can thus be disposed of to avoid deterioration of the overall statistics.

While *identification* based on a search for the true response that is closest to the measured response might be successful for very noisy responses, security applications typically require a fixed limit on the amount of incorrectly read bit positions to thwart guessing attacks. In an *authentication* scenario, an attacker might otherwise always be authenticated as the device that happens to be closest to the guess – an unacceptable issue. In a *key storage* scenario, the device is not in the possession of any true response, neither its own nor of any other device, thus matching as a method itself is out of scope. Instead, post processing such as a HDS, typically incorporating an ECC, is used to infer a sufficiently reliable estimation of the response provided by this device in a previous access. The fixed limit on the number of incorrectly read bits that can be corrected is then imposed by the choice of ECC. It should furthermore be not greater than necessary to achieve reliability goals, so implementation cost is kept low and the risk that error correction is strong enough to provide the same output for the same helper data on different devices is minimized. The consequences of such a fixed limit, set to e.g. half of the minimal HD between true responses, is depicted in the projections in Fig. 3.1a by dashed lines. In 12 out of 93 accesses, authentication or key reconstruction would fail.

Finally, the above concerned itself solely with the identifiability within the recorded dataset. Whether the conclusions drawn from this dataset regarding identifiability hold for a larger population of devices, other challenges, or even just another equally sized dataset recorded later, depends on the underlying probability distributions. The recorded dataset therefore needs to be representative and of sufficient size to allow to generalize the observed behavior. The question of dataset size is discussed in more detail in Sec. 3.7.

3.3. Interdependence and Overlap Among Metrics

Since all metrics outlined in the previous chapter are to measure either the reliability or unpredictability of a PUF candidate, it is expected behavior that they produce similarly good or bad grades for a given candidate. Some of the metrics, however, produce not just a similar grade, but are fully determined by the result of another metric. There are cases where such metrics are helpful, e.g. if they are designed to summarize a set of metrics in a single grade, but more often than not such metrics are proposed as independent metrics with no hint on the relationship. This section therefore explains the interdependence and overlap found among the introduced metrics.

3.3.1. MoD (Bit-Alias), Uniqueness, and Bitwise Entropy of Inter-Class Distribution

The common source for either version of Uniqueness, see (2.18) and (2.28), Bit-Alias (2.33), and bitwise entropy according to Maes (2.56), is the MoD. The abbreviation MoD is introduced because the corresponding mathematical symbol differs depending on the considered scenario, e.g. m_b for single-challenge PUFs as favored by Maiti et al. [70], $m_{k,l}$ in case of multiple identifiers per device as proposed by Hori et al. [72], or m_l for a single identifier per device as with Maes [82]. For an easy to grasp understanding of the MoD, note that it is the Bit-Alias for single-challenge PUFs.

The overlap between MoD and Maes' bitwise entropy metric $\overset{\text{Mb}}{H}$ becomes obvious when recalling that the entropy of independent Bernoulli RVs only depends on their success probabilities and Maes uses the MoD as an estimation for them. With this relationship, it becomes trivial to port bitwise entropy to other scenarios such as single-challenge PUFs, where the success probabilities would equal the Bit-Alias.

That either form of Uniqueness is fully determined by the MoD although it apparently is the mean of a set of HDs requires a more comprehensive explanation. As hinted at in [57, 72] and explicitly discussed in [**60**, 87], the order of operations for Uniqueness may be swapped – since summation is commutative – so that it may be seen as the mean among bit positions of a function of MoD. To visualize this for the version from [70], combine (2.2) with (2.18) into

$$\bar{\bar{h}} = \frac{2}{D(D-1)} \sum_{d=1}^{D-1} \sum_{d'=d+1}^{D} \left(\frac{1}{B} \sum_{b=1}^{B} \bar{x}_{b,d} \oplus \bar{x}_{b,d'} \right) \tag{3.1}$$

and change the order of summation

$$\bar{\bar{h}} = \frac{1}{B} \sum_{b=1}^{B} \left(\frac{2}{D(D-1)} \sum_{d=1}^{D-1} \sum_{d'=d+1}^{D} \bar{x}_{b,d} \oplus \bar{x}_{b,d'} \right). \tag{3.2}$$

For the version from [72], (2.30) can be rearranged to

$$\overset{\text{Hu}}{\bar{g}} = \frac{1}{L}\sum_{l=1}^{L}\left(\frac{1}{K}\sum_{k=1}^{K}\left(\frac{4}{D^2}\sum_{d=1}^{D-1}\sum_{d'=d+1}^{D}\bar{x}_{l,k,d}\oplus\bar{x}_{l,k,d'}\right)\right). \tag{3.3}$$

The innermost operation in both (3.2) and (3.3) is a sum of an XOR between all possible pairs of devices for the response bit position or identifier and identifier bit position selected by the outer summations. As mentioned in [72], the result of this operation is fully determined by the number of 1s encountered:

$$\sum_{d=1}^{D-1}\sum_{d'=d+1}^{D}\bar{x}_{l,k,d}\oplus\bar{x}_{l,k,d'} = \left(D-\sum_{d=1}^{D}\bar{x}_{l,k,d}\right)\sum_{d=1}^{D}\bar{x}_{l,k,d} \tag{3.4}$$

$$\sum_{d=1}^{D-1}\sum_{d'=d+1}^{D}\bar{x}_{b,d}\oplus\bar{x}_{b,d'} = \left(D-\sum_{d=1}^{D}\bar{x}_{b,d}\right)\sum_{d=1}^{D}\bar{x}_{b,d} \tag{3.5}$$

Using the definition of Bit-Alias, cf. (2.33), the latter may be rewritten as

$$\sum_{d=1}^{D-1}\sum_{d'=d+1}^{D}\bar{x}_{b,d}\oplus\bar{x}_{b,d'} = \left(D-Dm_b\right)Dm_b. \tag{3.6}$$

This equality can be intuitively reconstructed by once more considering the fact that summation is commutative. Hence the order of XORs may be chosen freely and so the order of bits in the vector established along the dimension of devices. This allows to arrange all bits equal to 0 at the left and all bits equal to 1 at the right hand side, cf. Fig. 3.2. Now all XORs in the 0 region, of which there are $\left(D-Dm_b\right)\left(D-Dm_b-1\right)/2$, and all XORs in the 1 region, count $Dm_b\left(Dm_b-1\right)/2$, yield zero, while the remaining XORs yield 1. Since the overall number of XORs performed is $D\left(D-1\right)/2$,

$$\begin{aligned}\sum_{d=1}^{D-1}\sum_{d'=d+1}^{D}\bar{x}_{b,d}\oplus\bar{x}_{b,d'} &= \frac{D\left(D-1\right)}{2}-\frac{\left(D-Dm_b\right)\left(D-Dm_b-1\right)}{2} \\ &\qquad -\frac{Dm_b\left(Dm_b-1\right)}{2} && (3.7)\\ &= \frac{2D^2m_b-2D^2m_b^2}{2} && (3.8)\\ &= D^2m_b\left(1-m_b\right). && (3.9)\end{aligned}$$

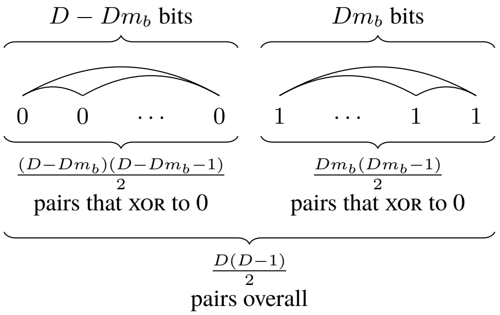

$D - Dm_b$ bits
Dm_b bits
0 0 ⋯ 0
1 ⋯ 1 1
$\frac{(D-Dm_b)(D-Dm_b-1)}{2}$ pairs that XOR to 0
$\frac{Dm_b(Dm_b-1)}{2}$ pairs that XOR to 0
$\frac{D(D-1)}{2}$ pairs overall

Figure 3.2.: Reordering makes obvious that the result of an XOR between all pairs of bits in a set of bits is only determined by the number of bits equal to 1.

Therefore, Uniqueness in either form is solely a function of MoD:

$$\bar{\bar{h}} = \frac{1}{B}\sum_{b=1}^{B}\left(\frac{2D}{D-1} m_b \left(1 - m_b\right)\right) \tag{3.10}$$

$$\overset{\text{Hu}}{\bar{g}} = \frac{1}{L}\sum_{l=1}^{L}\left(\frac{1}{K}\sum_{k=1}^{K} 4m_{l,k}\left(1 - m_{l,k}\right)\right) \tag{3.11}$$

The observation that bitwise entropy as well as Uniqueness are both aggregates of the MoD enables interesting findings about their expectation for an ideal PUF and their sensitivity to flaws. See sections 3.4.3 and 3.6 for details.

3.3.2. MoK and Diffuseness

Diffuseness according to Hori et al., see (2.27), is supposed to measure how unpredictable the identifiers produced from a certain device are. It is the mean HD between all possible pairs of identifiers produced by that device and is thus similar to Uniqueness, see (2.30). Due to this similarity, Diffuseness can be rewritten as a function of the mean along the axis of identifiers (MoK) [**60**],

$$\overset{\text{Hd}}{g}_d = \frac{1}{L}\sum_{l=1}^{L} 4m_{l,d}\left(1 - m_{l,d}\right), \tag{3.12}$$

where the MoK in Hori's scenario is

$$m_{l,d} = \frac{1}{K} \sum_{k=1}^{K} \bar{x}_{l,k,d}, \tag{3.13}$$

just as Uniqueness can be written as a function of MoD. The findings described in Sec. 3.6 thus also apply to Diffuseness, if the number of devices D is replaced by the number of identifiers K and the number of bit positions is L instead of KL.

3.3.3. Convergence of Metrics Through Multiple Means

The associativity and commutativity of the arithmetic mean, which has already been used e.g. to show that Uniqueness is a function of MoD, may also lead to convergence among initially distinct metrics. A simple example in this regard is mean Bit-Alias and mean Uniformity. Bit-Alias (2.33) is supposed to detect response bit positions that have a tendency to turn out the same value among all devices, whereas Uniformity (2.36) shall detect if there are devices that produce responses with very high or very low HW [70]. So both metrics have a meaningful intent and produce independent insight, since they operate on different dimensions of the dataset. However, both metrics produce not one grade but a set of results, precisely B Bit-Alias values m_b and D Uniformity values m_d, which make some authors only report the minimum, mean, and maximum of both. In such cases, the reported mean Bit-Alias has been equal to the mean Uniformity every time, but without this conspicuous fact ever been discussed. Based on this experience, it seems worthwhile to mention that mean Bit-Alias

$$m = \frac{1}{B} \sum_{b=1}^{B} m_b \qquad = \frac{1}{BD} \sum_{b=1}^{B} \sum_{d=1}^{D} \bar{x}_{b,d} \tag{3.14}$$

and mean Uniformity

$$m = \frac{1}{D} \sum_{d=1}^{D} m_d \qquad = \frac{1}{BD} \sum_{b=1}^{B} \sum_{d=1}^{D} \bar{x}_{b,d} \tag{3.15}$$

are mathematically equal and it is thus redundant to report both, even though one may disregard it a rather trivial finding.

3.3.4. MoA, Reliability, Bit Error Rate, Steadiness, Correctness, and Min-Entropy of Intra-Class Distribution

Among the metrics to measure the reliability of a PUF candidate exists a considerable group of metrics that are fully or almost fully defined by the mean along the axis of accesses (MoA). The MoA follows from (2.44) with the already known option to use l, k instead of b, c if the PUF candidate uses identifiers. Those that are almost fully defined have an additional dependency on the way true responses are defined, which may vary.

The metric Reliability according to Maiti et al. [70] is defined for single-challenge PUFs as the mean intra-class HD at certain environmental conditions, where each HD compares the response from one access to a device to that device's true response. Although it does not contain the eye catching double sum to take the mean of HD between all possible pairs along an axis, it *is* the mean of some HDs, so reordering of summations is again possible. For single-challenge PUFs as considered by Maiti et al., this yields

$$\overset{\text{II}}{h}_e = \frac{1}{AD}\sum_{a=1}^{A}\sum_{d=1}^{D}\left(\frac{1}{B}\sum_{b=1}^{B} x_{a,b,d,e} \oplus \bar{x}_{b,d,e}\right) \tag{3.16}$$

$$= \frac{1}{BD}\sum_{b=1}^{B}\sum_{d=1}^{D}\begin{cases} m_{b,d,e} & \text{if } \bar{x}_{b,d,e} = 0 \\ 1 - m_{b,d,e} & \text{if } \bar{x}_{b,d,e} = 1 \end{cases} \tag{3.17}$$

with $m_{b,d,e}$ from (2.44) with c omitted. Note that the cases are required to account for the various ways true responses can be chosen. If bit error rate is inferred from the mean rather than the maximum intra-class HD, cf. Sec. 2.5.3, it equals one of the $\overset{\text{II}}{h}_e$ and is thus also determined by the MoA. Thereby typical choices are the $\overset{\text{II}}{h}_e$ that corresponds to reference environmental conditions or the largest $\overset{\text{II}}{h}_e$ among the specified range of environmental conditions.

Correctness, see (2.26), similarly depends on the chosen true responses and may be rewritten as

$$\overset{\text{Hc}}{g}_d = 1 - \frac{2}{KL}\sum_{l=1}^{L}\sum_{k=1}^{K}\begin{cases} m_{l,k,d} & \text{if } \bar{x}_{l,k,d} = 0 \\ 1 - m_{l,k,d} & \text{if } \bar{x}_{l,k,d} = 1 \end{cases}, \tag{3.18}$$

where the MoA $m_{l,k,d}$ follows from (2.44) with the previously introduced l, k as replacement for b, c. Note that in this form it is more easy to spot that although the metrics by Hori et al. are supposed to be in the range zero to one, with the latter the optimal result, Correctness in fact has a range of -1 to 1, where the negative range is reached if too many of the identifier bits do not match their true response in too many accesses, e.g. due to operation at different environmental conditions.

For Steadiness, the fact that it is fully determined by the MoA is evident already from (2.61). Furthermore, it differs from two other metrics only by scaling of summands: The first is Correctness, if it has been calculated with true responses found by majority vote at the same environmental conditions, because the cases in

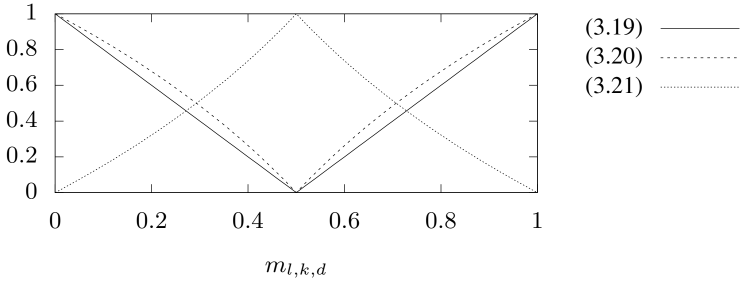


Figure 3.3.: Summands in (3.19), (3.20), and (3.21) over $m_{l,k,d}$.

(3.18) can then be replaced with a max operator as found in Steadiness [88]:

$$\overset{\text{Hc}}{g}_d = \frac{1}{KL} \sum_{l=1}^{L} \sum_{k=1}^{K} \left(-1 + 2 \max\left(m_{l,k,d}, 1 - m_{l,k,d}\right)\right) \tag{3.19}$$

$$\overset{\text{Hs}}{H}_d = \frac{1}{KL} \sum_{l=1}^{L} \sum_{k=1}^{K} \left(1 + \log_2 \max\left(m_{l,k,d}, 1 - m_{l,k,d}\right)\right) \tag{3.20}$$

The second is min-entropy of the intra-class distribution, see Sec. 2.6.2, if it is based on bit positions rather than byte positions as done by Holcomb et al.:

$$\overset{\text{a}}{H}_\infty = \sum_{l=1}^{L} \sum_{k=1}^{K} \left(-\log_2 \max\left(m_{l,k,d}, 1 - m_{l,k,d}\right)\right) \tag{3.21}$$

Note that for this metric, smaller values are better, and an ideal PUF produces zero. The difference in scaling of the summands in (3.19) through (3.21) over $m_{l,k,d}$ is visualized in Fig. 3.3.

In summary, it is sufficient to report only one of these metrics. Preferable one that depends on the true responses, since this also detects stable incorrect bit positions and automatically coincides with Steadiness if there are none, i.e. if the true response matches the response in the majority of accesses.

3.4. Risk of Blind Spots by Excessive Use of Means

The pursuit of a single scalar to grade the unpredictability of a PUF candidate despite the high dimensionality of PUF data creates a demand for dimensionality reduction. One of the most easy ways to perform this while still apparently considering all data in the calculation is to take the mean along a dimension to be reduced. However, this may lead to blind spots, because different statistical defects may cancel each other out while summing up. Three examples for this behavior will be given in the following.

3.4.1. Randomness

Randomness as proposed by Hori et al. [72], cf. (2.60), is supposed to measure the overall amount of min-entropy in the dataset. As most metrics by Hori et al., it provides one result per device, here the min-entropy based on the mean number of response bits that turned out as 1 among all accesses to a device. Note that although the metric is targeted towards unpredictability and not reliability, it does not operate on true responses, but considers all accesses. Furthermore, in conrast to e.g. Steadiness, the $\max$ operator and logarithm are applied *after* taking the mean along accesses, identifiers, and identifier bit positions. This has two major implications.

First, PUF candidates that produce obviously flawed identifiers may still receive good grades regardless of their reliability as long as the overall ratio of 0s and 1s is balanced. This includes e.g. devices that produce all-0 and all-1 identifiers in the same quantity and devices where equal amounts of bit positions are stuck at 0 and stuck at 1. Fig. 3.4 visualizes these and some other examples. In these examples, the mean allows for too many 1s in some identifiers or bit positions to cancel out with too few 1s in others. Randomness can therefore only flag PUF candidates that contain unbalanced flaws, such as a global bias towards 0 or 1.

Second, even PUF candidates with unbalanced flaws may not cause bad results for Randomness if they additionally are unreliable. This stems from taking the mean among all accesses rather than operating on true responses respectively true identifiers. The effect is demonstrated in Fig. 3.5, which plots the achieved Randomness over the relative amount of 1s in the true identifiers and the relative amount of bit flips among all accesses. To calculate this, let

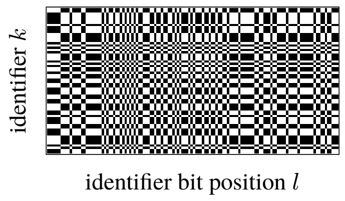
$$m_d{}' = \frac{1}{AKL} \sum_{l=1}^{L} \sum_{k=1}^{K} \sum_{a=1}^{A} x_{a,l,k,d}$$

from (2.60) be rephrased as

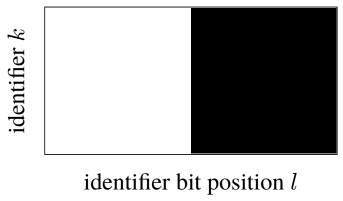
$$m_d{}' = m_d + \delta \tag{3.22}$$

to separate balance in true identifiers from reliability. Then

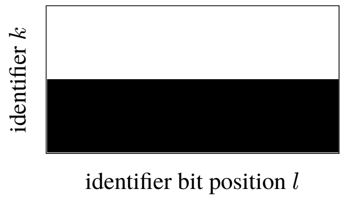
$$m_d = \frac{1}{KL} \sum_{l=1}^{L} \sum_{k=1}^{K} \bar{x}_{l,k,d} \tag{3.23}$$



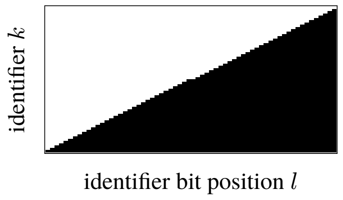
(a) All identifiers are either equal to or the inverse of a single pattern. Noticeable as every line always changes at the same columns.



(b) Equal numbers of bit positions are stuck at 1 and 0.



(c) All-0 and all-1 identifiers occur equally often.



(d) Identifiers represent their index in a thermometer code (unary coding)

Figure 3.4.: Examples of simulated devices that produce obviously flawed sets of identifiers but still achieve the best result in Randomness according to [72]. Devices are assumed to be noise free in these examples, i.e. they produce the same identifiers in all accesses.

is the FHW of, i.e. the relative amount of 1s in, the set of true identifiers and δ is the change caused by bit errors during accesses. Let further t denote the relative amount of bit errors during all accesses and those bit errors be located randomly, so the probability of a bit flip from 1 to 0 depends on the relative amount of 1s in the response and the probability of a bit flip from 0 to 1 depends on the relative amount of 0s. Now

$$\delta^{\downarrow} = t m_d \tag{3.24}$$

approximates the relative amount of bit flips from 1 to 0,

$$\delta^{\uparrow} = t(1 - m_d) \tag{3.25}$$

approximates the relative amount of bit flips from 0 to 1, and

$$\begin{align} \delta &= \delta^{\uparrow} - \delta^{\downarrow} \tag{3.26} \\ &= t(1 - 2m_d). \tag{3.27} \end{align}$$

The last equation tells that randomly located bit errors tend to balance the amount of 0s and 1s in an identifier: If m_d is below 0.5, so an imbalance towards 0s, the factor in parentheses becomes positive, and δ increases the FHW. If m_d is above 0.5, so an imbalance towards 1s, the factor in parentheses becomes negative, and δ decreases the FHW. Furthermore, combination with (3.22) provides

$${m_d}' = t + m_d(1 - 2t), \tag{3.28}$$

which shows that while the relative amount of bit flips t approaches 0.5, the amount of bias m_d within true identifiers becomes less relevant to the outcome of the metric. If bit errors are not randomly located, then $\delta \in [\max(-t, -m_d), \min(t, 1 - m_d)]$ for two reasons: The limits $-t$ and t are reached if all bit errors are of the same type. The limits $-m_d$ and $1 - m_d$ reflect that the FHW cannot be changed towards outside $[0, 1]$, i.e. there cannot be more bit flips of one type than such bits are available, e.g. not more 1 to 0 bit flips than 1 bits in the true response. In either case there is a non negligible risk that the noise in a PUF candidate's output incorrectly improves the result for Randomness despite the true identifiers being heavily biased.

After all, a good RNG would also achieve the optimal result for Randomness and it requires additional metrics such as Steadiness to distinguish an RNG from a PUF. Although the name Randomness may be interpreted in a sense that this is intended behavior, its description in [72] rather suggests that the metric is supposed to flag PUF candidates that produce biased and therefore easier to predict responses, which it may fail to do when the PUF candidate is too noisy. This behavior is thus an important finding to consider when using the metric.

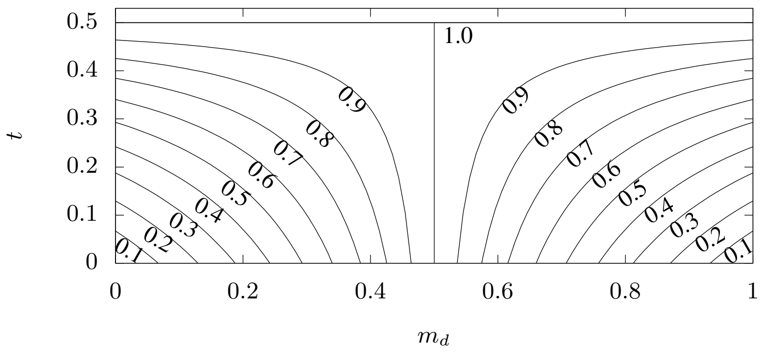


Figure 3.5.: Contour plot of Randomness over relative amount m_d of 1s in true identifiers, and relative amount t of randomly located bit errors. Note that since true identifiers are obtained in [72] by majority vote among all samples, $0 \leq t \leq 0.5$.

3.4.2. Mean Bit-Alias

A very related issue to that depicted in Fig. 3.4 arises when – instead of all B Bit-Alias, or in general MoD values – only the mean Bit-Alias is reported. Even if this value is very close to the optimal 0.5, it provides no evidence that all – or just any – response bit positions are unpredictable. The reason is that a bias towards 1 on some bit positions may cancel out with a bias towards 0 at other bit positions. It might in fact be that a heatmap of the responses from all devices looks similar to one of the subfigures in Fig. 3.4 with identifier bit position replaced by response bit position and identifiers replaced by devices. Just as described above for Randomness, the mean Bit-Alias for all plots shown in Fig. 3.4 would be optimal, since the statistical defects are balanced.

Although reporting all MoD values individually is advisable for a proper performance evaluation, there can be situations where more compressed information is sufficient. For example, reporting the minimum and maximum values can be sufficient if both are close enough to 0.5. At this point, however, it should be noted that even then the numbers may fail to identify a flaw if they are based on too few devices, see Sec. 3.7.

3.4.3. Uniqueness

Another example where taking the mean inadvertently discards important information is Uniqueness, where the version according to Maiti et al. [70] is similarly affected as the version according to Hori et al. [72]. The difference between them is twofold. First, the version by Maiti et al. operates on responses of single-challenge PUFs whereas the version by Hori et al. operates on identifiers constructed from multi-challenge PUFs. The second difference lies in normalization. Apart from these differences, both report the mean HD among devices, either based on the response or all identifiers produced by a device.

To take the mean of HDs rather than raw data creates the impression of considering how all the bits of response respectively identifier change as a whole, i.e. to capture correlations among response bits, which is claimed in e.g. [15, 75, 89]. However, this could possibly work only if the entire distribution of HD values is reported, and not just the mean. The reason is that the mean introduces another summation that can be reordered to occur *within* the HD operation, more precisely after the XOR but before the summation of bit positions, cf. Sec. 3.3.1. Although in this case bias towards 0 and bias towards 1 cannot cancel out between bit positions due to the involved quadratic function of MoD, the fact that it can be expressed as a function of MoD proves that correlations or other dependencies cannot be reflected by Uniqueness. Consequently, Uniqueness should be perceived not as a metric of its own, but rather as an aggregate of MoD.

Other aggregates of MoD are the mean Bit-Alias or the bitwise entropy $\overset{\text{Mb}}{H}$ by Maes, see (2.56). Among these MoD aggregates, the mean Bit-Alias is once again a poor choice due to the blind spot discussed in Sec. 3.4.2. Uniqueness and bitwise entropy are not prone to this, because they take the mean among bit positions of some nonlinear function of the MoD, so that any bias causes a reduction of that bit position's contribution to the sum regardless of whether that bias is towards 0 or towards 1. A more detailed discussion which of these nonlinear aggregates is the best choice can be found in Sec. 3.6.

3.5. Statistical Modeling of PUF Outputs

Many of the metrics outlined in the previous chapter do not choose a statistical model for the PUF, in particular the HD and HW based metrics, which merely calculate some mean along some axes. However, as will be shown in the following sections of this chapter, a suitable statistical model is a necessity to judge whether the sample mean calculated by some of these metrics is acceptable, reason for concern, or possibly meaningless due to a too small sample size. Without statistical model, no CIs can be calculated, nor any hypothesis test be performed.

The model does not necessarily have to reflect the actual physical processes within the physical object when it is evaluated, but capture the statistical properties of the produced responses or identifiers with sufficient accuracy, and be simple enough to construct hypothesis tests, CIs, etc. Separate analysis has to ensure that the PUF output is not the result of a sophisticated algorithm, as discussed already in Sec. 1.6, but stems from actual physical processes, which is canonically the only source of randomness admitted the label *true* randomness. A counter example in this regard would be a fake PUF built from e.g. an advanced encryption standard (AES) implementation with a device specific key, the challenge as plain text and the cipher text as response. Although this fake PUF would feature excellent statistical properties, an entity that knows the device specific key could predict the output for any challenge, so the PUF would possess no entropy at all. If the connection between physical processes and PUF output is established, though, the pass of statistical tests can show that the intended physical sources of randomness make a sufficient amount of entropy measurable and it is properly extracted and post-processed.

In the remainder of this section, three suitable statistical models are presented. First, the univariate Bernoulli model, which is the most simple from a mathematical point of view, but the most restrictive from a practical point of view, since it is only applicable to very homogeneous PUF designs. Among the few existing metrics that do choose a statistical model for the PUF, most choose this model. Second, the multivariate Bernoulli model, which has been used in e.g. [**71**, 82]. It is mathematically more difficult to use than a univariate model, but allows to reflect e.g. non-homogeneous Bit-Alias. Third, a multivariate categorical model, which has been used to analyze the intra-class distribution by Holcomb et al. [78], but is an original contribution for the purposes shown in this work, such as the analysis of the inter-class distribution. Before the unpredictability of a PUF can be analyzed using one of the above mentioned models, however, the run-time noise and the effects of environmental conditions have to be modeled and compensated for to obtain samples of the true responses related to the manufacturing variations of the physical object.

3.5.1. Modeling Run-Time Noise

The most flexible model for run-time noise is that of individual binary symmetric channels (BSCs) for each bit position, challenge, device, and environment. So the response obtained by an access to the PUF is the output of such channel with the true response as input, where the channel parameter may be different for each channel. The model does not imply a specific way to define the true response and therefore supports common choices such as majority vote or selection of the first sample, both at reference environmental conditions. The probability of a bit flip can then be estimated from available accesses at the respective environmental conditions. An estimated bit flip probability of larger than 0.5 indicates that the *expected* response, i.e. the expectation at the output of the channel, differs from the true response. The latter may occur e.g. due to a change in environmental conditions or because a single access was used to define the true response and that happened to be different from most others. Since a BSC with bit flip probability $\tilde{p}$ is statistically equivalent to an XOR with a $\mathfrak{B}(1, \tilde{p})$ distributed RV, above model can be written as

$$x_{a,b,c,d,e} = \bar{x}_{b,c,d} \oplus \tilde{x}_{a,b,c,d,e}, \tag{3.29}$$

where $\tilde{x}_{a,b,c,d,e}$ are realizations of the RV $\tilde{X}_{b,c,d,e}$, written as

$$\tilde{x}_{a,b,c,d,e} \leftarrow \tilde{X}_{b,c,d,e} \tag{3.30}$$

and the Bernoulli distribution with parameter $\tilde{p}_{b,c,d,e}$ of $\tilde{X}_{b,c,d,e}$ written as

$$\tilde{X}_{b,c,d,e} \sim \mathfrak{B}(1, \tilde{p}_{b,c,d,e}), \tag{3.31}$$

which is equivalent to the PMF

$$\mathrm{P}\left(\tilde{X}_{b,c,d,e} = o\right) = \begin{cases} \tilde{p}_{b,c,d,e} & o = 1 \\ 1 - \tilde{p}_{b,c,d,e} & o = 0 \\ 0 & \text{otherwise.} \end{cases} \tag{3.32}$$

Alternatively, one may place a deterministic transformation before the channels to account for the effects of environment and view the expected response in a given environment as input to the channel. Either way provides the necessary relationship between the observable response data $x_{a,b,c,d,e}$ and the true responses $\bar{x}_{b,c,d}$ that are modeled in the following subsections.

3.5.2. Univariate Bernoulli Model

The univariate Bernoulli model assumes that all true responses consist of bits from a memoryless Bernoulli source, i.e. are IID. In other words, the true response for every bit position, challenge, and device is found by an independent draw from the same $\mathfrak{B}(1, p)$ RV:

$$\bar{x}_{b,c,d} \leftarrow X \sim \mathfrak{B}(1, p) \tag{3.33}$$

This model is also the standard model for TRNG tests, and fits an ideal PUF if $p = 0.5$ is chosen, since then every element of the response space is equiprobable, given a response space of 2^B elements, each being a string of B bits. The combination of simplicity and the ability to match an ideal PUF makes the model a desirable choice.

The model parameter p is either given by theory, in which case it is usually chosen to $p = 0.5$, or estimated from the observed PUF responses. For the latter case, m from (2.42) is a uniform minimum variance unbiased (UMVU) estimator for p. In this model, all true response bits, i.e. the product of the number of bit positions B, challenges C, and devices D, can be used to estimate a single parameter. Because these bits all constitute independent drawings from the same RV, the estimation comes with smaller, i.e. better, CIS than the following models for any given dataset size.

For many metrics, this model can be identified as an implicit assumption, because they calculate their result among all response bits regardless of bit position, challenge, etc. The caveat that comes with this simple model, however, is that the assumption of independent drawings from the same source may not hold in practice, so the model oversimplifies. This can cause metrics to overestimate e.g. the unpredictability of a PUF candidate. An overall balance of 0s and 1s, for example, would cause p to be estimated close to the ideal 0.5, even if individual response bit positions are severly biased and thus easy to guess for an attacker. To rely solely on the esimtated p would therefore jeopardize the security of the whole application. The use of metrics based on this model thus requires to validate its applicability first, e.g. through an appropriate statistical hypothesis test.

For statistical hypothesis tests, the simplicity of this model makes it easy to infer the expected distribution of the test statistic under the null hypothesis. It needs to be considered, though, that any test based on this model also includes the assumption of IID response bits, although it may be aimed to detect other flaws such as dependence. An example for this is a test for dependence between two response bit positions that compares the observed number of 00, 01, 10, and 11 combinations to $(1-p)^2$, $(1-p)p$, $(1-p)p$, and p^2. If the counts are sufficiently different from the expected shares, the test may conclude dependence, although the difference may be solely due to a different probability for a 1 among the bit positions. To distinguish the effect of dependence from a difference in probability for a 1 requires a test that avoids the latter assumption, which leads to the multivariate model described in the following.

3.5.3. Multivariate Bernoulli Model

A model that tolerates more flaws without being invalidated is the independent multivariate Bernoulli model. One of these flaws is the frequently encountered issue where certain bit positions or challenges have a higher probability to produce a 1 than a 0, while other bit positions or challenges favor the other value or are unbiased. Such behavior has been shown for example in [82] for a latch, a buskeeper, an arbiter, and an RO PUF design, in [**71**, **85**] for another RO PUF, and in [**74**] for an SRAM PUF. It may be caused e.g. by imperfections in the layout, unequal routing on an FPGA, or manufacturing variations in the photolithography masks, which shift the expectations for the manufacturing variations in different ways so each bit position or challenge tends towards either 0 or 1 among devices by an individual amount. The exact reason for such a bias is only of interest to a designer that aims to improve the PUF circuit design, though. For a proper evaluation, it is sufficient that the statistical model can represent such imperfections so they do not cancel each other out, which may occur under the univariate Bernoulli model as described in Sec. 3.5.2.

The multivariate Bernoulli model assumes that the true responses are drawn from a Bernoulli random vector $\underline{X} \sim \mathfrak{B}(1, \underline{p})$, where $\underline{p}$ contains elements p_λ with $\lambda \in \{1, \ldots, \Lambda\}$. The necessary length Λ of the random vector depends on the PUF candidate to be modeled. In the most intuitive form, a single-challenge PUF is modeled by one RV per response bit position, in which case $\Lambda = B$ and

$$\underline{\bar{x}}_d \leftarrow \underline{X} \sim \mathfrak{B}(1, \underline{p}). \tag{3.34}$$

This formula also holds for the case where each device produces one identifier and each bit position of the identifier is modeled by its own Bernoulli RV, in which case $\Lambda = L$. Note that arbitrary mappings between tuples (b, c) or (k, l) and λ are possible to suit the PUF candidate to be modeled. Furthermore estimation of $\underline{p}$ may be simplified by additional assumptions such as equality among parts of $\underline{p}$.

Since the benefit of the multivariate Bernoulli model over the univariate Bernoulli model is the ability to represent varying bias, the parameter vector $\underline{p}$ is usually estimated from the dataset. For the exemplary case of one RV per response bit position of a single-challenge PUF, the vector of Bit-Alias, m_b $b \in \{1, \ldots, B\}$ cf. (2.33), is an UMVU estimator for $\underline{p}$. Similarly for the identifier based example, the $m_{b,c}$ from (2.32) constitute UMVU estimators for the elements of $\underline{p}$, which just need to be reordered depending on how identifiers are constructed from responses to multiple challenges.

In both cases, only D samples are available to estimate each of the B respectively BC parameters, instead of BD respectively BCD samples as with the univariate Bernoulli model. Although this fewer number of available observations leads to larger CIs, i.e. probably less accurate estimation of parameters, it likely produces a more accurate overall assessment of unpredictability for PUF designs with varying bias, because their flaws are less likely to cancel each other out as with the univariate Bernoulli model. In general, the selection of a model shall always be guided by the underlying real-world mechanisms, so the statistical model is a *reasonable* simplification of them. The solution to this issue must therefore be an increased number of sample devices for analysis instead of a change in model.

Since the multivariate Bernoulli model used in this work maintains an assumption of independence between the RVs, it allows to represent varying bias among challenges or bit positions, but cannot represent correlations. Modeling all possible correlations between two or more RVs would render the model unusable, since the number of parameters would be infeasible to estimate. To limit the amount of modeled correlations to e.g. pairs of two RVs would reduce the number of parameters, but is an option not considered in this work in favor of the multivariate categorical model. The multivariate categorical model is able to represent not just correlations but arbitrary dependencies, even nonlinear ones, as long as they affect only small groups of challenges or bit positions. It is explained in the following subsection.

3.5.4. Multivariate Categorical Model

For PUF designs that quantize the manufacturing variations in multi-bit symbols or where PUF cells interact with each other, certain response bit positions might become correlated. Such correlations may be exploited by an attacker to better predict the response of a device under attack, which means that they have to be considered in an assessment of unpredictability. In this situation, a dependent multivariate Bernoulli model may come to mind. However, a model of Λ dependent Bernoulli RVs has $2^\Lambda - 1$ parameters: Λ marginal success parameters, $\binom{\Lambda}{2}$ parameters for pairwise dependence, $\binom{\Lambda}{3}$ parameters for dependence among groups of three, and so on [90]. It is the same number of parameters a categorical distribution with 2^Λ categories requires, if the second axiom of probability is used, i.e. that the sum of probabilities over all outcomes equals unity. So instead of a dependent multivariate Bernoulli distribution, one may as well use a univariate categorical distribution with e.g. the entire response vector as an element of the response space. The sole difference is the meaning of the parameters. In the most simple example of $\Lambda = 2$, a multivariate Bernoulli distribution is parameterized by p_1 and p_2, the probabilities that the first and second bit, respectively, turn out to be 1, and their covariance σ_{12}, whereas the categorical distribution takes the parameters p_{00}, p_{01}, p_{10}, and p_{11}, the probabilities that both bits turn out as 00, 01, 10, and 11, respectively, of which one is redundant due to the second axiom of probability. Formulae to translate between both descriptions for arbitrary Λ can be found in [90].

On the one hand, a univariate categorical distribution would be the best choice from a scientific point of view, because it requires least assumptions. It would also fit best to the random oracle model, since it does not limit the response space to binary vectors, but supports arbitrary countable sets of answers. On the other hand, a straight forward estimation of the probability of each possible element of the response space must be infeasible for a PUF, otherwise the response space can be enumerated in reasonable time and the PUF is prone to a trivial brute-force attack. A univariate categorical model is thus impractically complex, so it is necessary to limit the number of modeled dependencies, e.g. to the strongest ones, to keep the complexity on a feasible level.

An example of this for single-challenge PUFs is to split the response vector $\underline{x}$ into Λ subresponses $\underline{z}_\lambda$ and model every subresponse by its own categorical distributed RV. This way the subresponses are still assumed independent of each other, but any dependencies among the response bit positions within a subresponse are fully modeled. Note that the way the response vector is split into subresponses is arbitrary, so a subresponse does not necessarily hold consecutive response bit positions. Even the number of absorbed response bit positions may differ between subresponses. In this multivariate categorical model, the parameterization ends up being a vector of functions that associate the elements of the respective set of outcomes to their respective probabilities:

$$\bar{\underline{z}}_{\lambda,d} \leftarrow Z_\lambda \sim \mathfrak{C}(\mathcal{Z}_\lambda \mapsto \underline{p}_\lambda) \tag{3.35}$$

$$\left(\bar{\underline{z}}_{\lambda=1,d} \quad \cdots \quad \bar{\underline{z}}_{\lambda=\Lambda,d}\right) \leftarrow \underline{Z} \sim \mathfrak{C}\left(\underline{\mathcal{Z} \mapsto \underline{p}}\right) \tag{3.36}$$

Estimation of the parameter vector $\underline{p}_\lambda$ for any particular RV is possible once more from the empirically observed sample counts. For example, if categorical RV Z_λ represents the response bit positions 1, 4, and 7, of a single-challenge PUF to be assessed, the eight elements $p_{\lambda,i}$ of its parameter vector $\underline{p}_\lambda$ may be estimated by

$$p'_{\lambda,1} = \frac{1}{D}\sum_{d=1}^{D}\begin{cases}1 & \text{if } \bar{x}_{b=1,d} = 0 \wedge \bar{x}_{b=4,d} = 0 \wedge \bar{x}_{b=7,d} = 0\\ 0 & \text{otherwise}\end{cases}, \tag{3.37a}$$

$$p'_{\lambda,2} = \frac{1}{D}\sum_{d=1}^{D}\begin{cases}1 & \text{if } \bar{x}_{b=1,d} = 0 \wedge \bar{x}_{b=4,d} = 0 \wedge \bar{x}_{b=7,d} = 1\\ 0 & \text{otherwise}\end{cases}, \tag{3.37b}$$

$$p'_{\lambda,3} = \frac{1}{D}\sum_{d=1}^{D}\begin{cases}1 & \text{if } \bar{x}_{b=1,d} = 0 \wedge \bar{x}_{b=4,d} = 1 \wedge \bar{x}_{b=7,d} = 0\\ 0 & \text{otherwise}\end{cases}, \tag{3.37c}$$

$$\vdots$$

$$p'_{\lambda,8} = \frac{1}{D}\sum_{d=1}^{D}\begin{cases}1 & \text{if } \bar{x}_{b=1,d} = 1 \wedge \bar{x}_{b=4,d} = 1 \wedge \bar{x}_{b=7,d} = 1\\ 0 & \text{otherwise}\end{cases}. \tag{3.37d}$$

3.5.5. Application of Statistical Models to Machine Learning Attacks

Soon after the invention of multi-challenge PUFs, modeling attacks using ML were introduced [23], which aim to predict the response of a particular device to yet unseen challenges after being trained with a sufficient number of CRPs from that particular device. However, predictability is an issue not just between CRPs on the same device. Predictability of PUF outputs is a multi-dimensional question, since any of the five dimensions introduced in Sec. 1.8 may serve as an axis for prediction. Among these five dimensions, predictability along samples and environmental conditions is a key feature, since this distinguishes PUFS from RNGS and allows to recognize a previously seen device for authentication, or recreate a key from some helper data. In contrast, it is a flaw if predictability arises among response bit positions, challenges, or devices.

The most intuitive application of the previously introduced models may be prediction along devices. So an attacker may obtain a sufficient number of devices of the same type as the device under attack and investigate their responses, which she is assumed able to since she is the legitimate customer of these devices. The so produced dataset can then be used to estimate the parameters of any of the above models, which might then allow to make intelligent guesses on the response of the device under attack. An example would be to observe that certain challenges produce a 0 at certain response bit positions on nearly all devices, which suggests the use of the multivariate Bernoulli model. The correspondingly low $p_{b,c}$ at these positions for these challenges would then recommend to guess these response bit positions as 0 if one of these challenges is applied, which likely increases the chance of a correct guess.

The statistical models are equally applicable to ML attacks, though, as Maes [82] showed for the univariate Bernoulli model. The typical scenario for ML attacks on PUFS is prediction along challenges, i.e. a ML model is trained with a sample set of CRPS from the device under attack and learns the relationships between challenge bits and response bits. Once training is complete, the ML model can be fed a yet unseen challenge and produces a prediction for the response based on the learned relationships. The unpredictability of the PUF candidate can then be estimated using the probability that the prediction is correct, cf. Sec. 2.5.2 and (2.58). So the parameters p of the univariate Bernoulli model respectively $\underline{p}$ of the multivariate Bernoulli model are not related to the probability of a 1 in the response, but the *probability of a correct guess* of the ML model. Which of the two models is appropriate depends on the PUF to be attacked and the model. As mentioned on p. 24 in Sec. 1.2.4, in particular for early arbiter PUF designs the predictions are correct with consistently high probability already after little amount of training. In this case the univariate Bernoulli model is an appropriate and efficient choice. Improved PUF designs may be less easy to learn by a ML model. For example, it may occur that the response to challenges with the highly influential bits [25] matching the pattern in the training CRPS can be predicted with higher accuracy than for other challenges. To evaluate ML models for such PUF candidates, the multivariate Bernoulli model can be used, where a RV represents a subset of challenges and response bit positions that can be predicted with approximately the same probability of being correct.

3.6. Optimum and Sensitivity of Uniqueness and Other Aggregates of MoD Under Univariate Bernoulli Model

3.6.1. Optimum of Uniqueness

Maiti et al. [70] claimed the optimum of their version of Uniqueness to be 0.5 without mathematical proof and many authors who used the metric since then adopted this claim without providing a proof either. However, the fact that Hori et al. [72] scaled their version of Uniqueness based on the *maximum* possible value and Maiti et al. showed in [57] that the *maximum* of their version of Uniqueness is *not* 0.5 is a contradiction that has not been addressed yet. This section therefore aims to clear up this contradiction.

The *maximum* of either version of Uniqueness is rather trivial to verify: Inspecting (3.10) shows that it is maximized if all its summands are maximized and that it equals any of its summands if they are equal. The summands are a quadratic function in their respective Bit-Alias m_b, and that function in vertex form equals

$$-\frac{2D}{D-1}\left(m_b-\frac{1}{2}\right)^2+\frac{D}{2(D-1)}. \tag{3.38}$$

The vertex form of a quadratic function allows to directly read out the location of maximum – or minimum if the coefficient of the quadratic term is positive. Here, the maximum is $D/2(D-1)$, which is achieved if m_b equals $1/2$. Since this applies to all bit positions b identically, $D/2(D-1)$ is proved as the maximum of Uniqueness according to Maiti et al., and it is achieved if all Bit-Alias equal $1/2$. Likewise, the maximum for Uniqueness according to Hori et al., claimed unity in [72], can be verified by rewriting the innermost summands of (3.11) in their vertex form

$$-4\left(m_b-\frac{1}{2}\right)^2+1. \tag{3.39}$$

This proves a maximum of unity, which is achieved if every bit position in every identifier turned out 0 as often as it turned out 1 among all devices in the dataset.

However, the *maximum* of the respective function is not necessarily the *optimum* with regard to unpredictability. To obtain the latter, one may use a statistical model of an ideal PUF such as the univariate Bernoulli model with $p = 0.5$. In this model all response bits, thus also all identifier bits, are IID. So the MoD follows a binomial distribution with success probability $p = 0.5$, D draws, and with support scaled from $\{0, 1, \ldots, D\}$ to $\{0, {}^1\!/\!_D, \ldots, 1\}$:

$$m_b \leftarrow M \tag{3.40}$$

$$\mathrm{P}(M = m_b) = \begin{cases} \binom{D}{Dm_b} 0.5^D & \text{if } m_b \in \{0, \frac{1}{D}, \ldots, 1\} \\ 0 & \text{otherwise} \end{cases} \tag{3.41}$$

The expectation of this RV is easily verified to be $1/2$ by considering the fact that a canonical binomial RV with D trials and success probability 0.5 has expectation $D/2$ and the scaled support divides this by D due to linearity of expectation. This, however, does not prove $m_b = 0.5$ to correspond to the optimum with regard to unpredictability, though, because either form of Uniqueness is a nonlinear function of m_b and thus linearity of expectation is not applicable. Instead, the so obtained PMF, which is the same for every bit position in every identifier, but depends on D, has to be further processed by considering the quadratic function in (3.10), (3.11). The PMF of any of their summands then is

$$\mathrm{P}(M' = m_b') = \begin{cases} \binom{D}{D/2} 0.5^D & \text{if } m_b' = \frac{D}{2(D-1)}, \\ \binom{D}{i} 0.5^{D-1} & \text{if } i \in \{0, 1, \ldots, \lfloor \frac{D-1}{2} \rfloor\}, \\ 0 & \text{otherwise} \end{cases} \tag{3.42}$$

for the version by Maiti et al., where

$$\begin{aligned} i &= \frac{D}{2} - D\sqrt{\frac{1}{4} - m_b' \frac{D-1}{2D}}, \\ m_b' &= \frac{2D}{D-1} m_b \left(1 - m_b\right), \end{aligned} \tag{3.43}$$

respectively

$$\mathrm{P}(M'' = m_{l,k}'') = \begin{cases} \binom{D}{D/2} 0.5^D & \text{if } m_{l,k}'' = 1, \\ \binom{D}{i} 0.5^{D-1} & \text{if } i \in \{0, 1, \ldots, \lfloor \frac{D-1}{2} \rfloor\}, \\ 0 & \text{otherwise} \end{cases} \tag{3.44}$$

for the version by Hori et al., where

$$\begin{aligned} i &= \frac{D}{2} - D\sqrt{\frac{1 - m_{l,k}''}{4}}, \\ m_{l,k}'' &= 4 m_{l,k} \left(1 - m_{l,k}\right). \end{aligned} \tag{3.45}$$

The formulae are obtained by mapping each possible value of m'_b respectively $m''_{l,k}$ to the corresponding number of devices i that produced a 1, which can then be used to calculate the probability that this value of the summand occurs. The first case in (3.42) and (3.44) can only occur if D is even.

Fig. 3.6 displays the PMFs for several values of D together with their expectation and standard deviation. In each of the four subfigures, the PMF is visualized as a series of arrows where the location on the x-axis shows the possible value of the RV and the height of the arrow indicates the corresponding probability. Because both versions of Uniqueness are quadratic functions of the MoD with different scaling, they can be represented in the same plot for a given D, where the bottom x-axis corresponds to the version by Hori et al. and the top x-axis to the version by Maiti et al. In the uppermost plot, the corresponding number of devices that produced a 1, i, is additionally printed next to each arrow. The PMFs look similar to the left half of a binomial PMF, because the MoD is binomial distributed and the quadratic functions effectively just add the right half of the distribution onto the left half, so e.g. for $D = 10$, $i = 1$ and $i = 9$ produce the same value for a summand. Compared to a binomial distribution, this doubles the probability, so the height of an arrow, for all values except the one that corresponds to exactly half of the devices producing a 1, which can only occur for even D. Finally, the expectation and standard deviation of the PMF is shown in each subfigure as a dashed vertical line and dashed horizontal arrow, respectively.

As visible from Fig. 3.6, the expectation for any summand in the version by Maiti et al. remains at 0.5 independent of the number of devices D in the dataset. In contrast, the expectation of the summands in the version by Hori et al. depends on D, although their maximum did not. The reason is that due to the quadratic function, the expectation no longer scales linearly with the number of trials, as it would for a binomial distribution. For the version by Maiti et al., the scaling exactly compensates this change in expectation caused by D, as it also depends on D, whereas for the version by Hori et al., its fixed scaling cannot compensate the change.

Given the expectation for a summand in either version of Uniqueness, the expectation of the overall function is easily found as the remaining operations are to take the mean among response bit positions respectively identifier bit positions and identifiers, which are linear operations on, for an ideal PUF, IID RVs. The expectation of either version of Uniqueness for an ideal PUF is thus equal to the expectation of any of its summands by linearity of expectation.

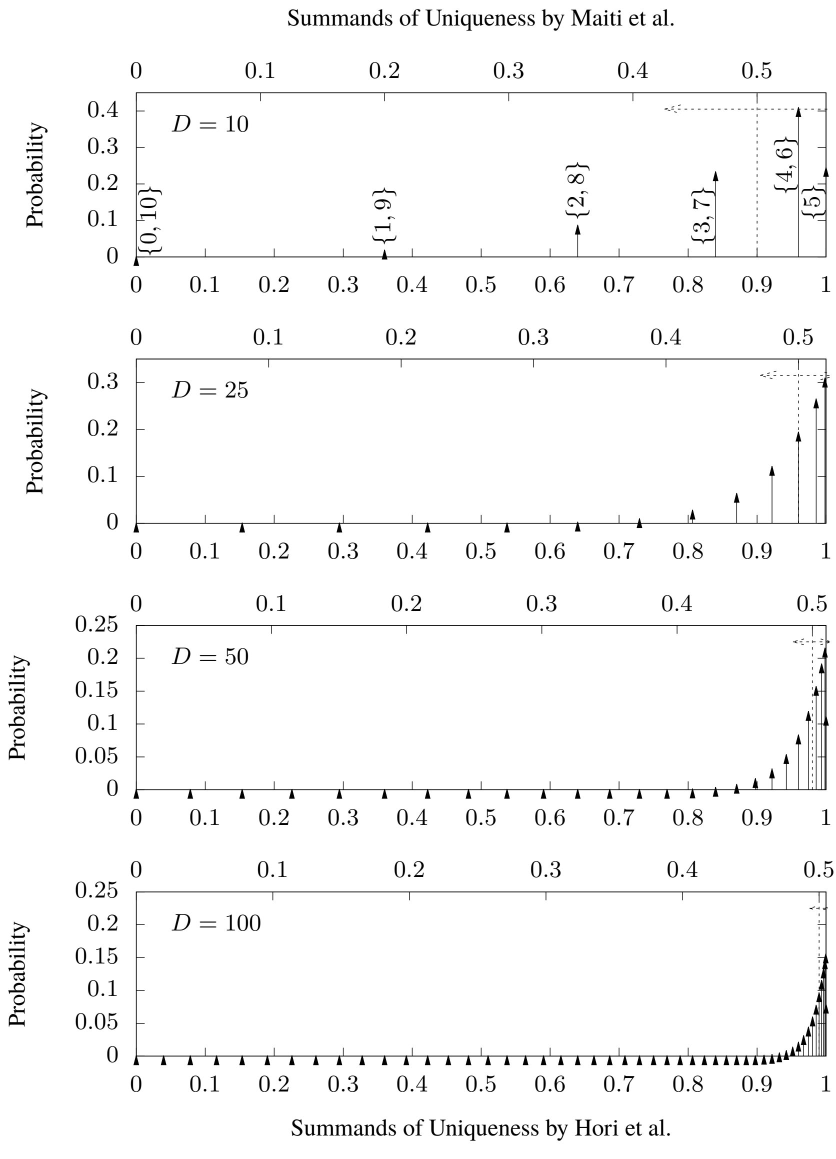


Figure 3.6.: PMFs of a summand in (3.10) (top x-axis) and (3.11) (bottom x-axis) for an ideal PUF and several values of D. Dashed lines indicate expectation and standard deviation of the PMFs. The uppermost plot also shows the number of devices that need to produce a 1 at a given response bit position to produce the respective value for a summand.

3.6.2. Optimum of Other Aggregates of MoD

The observation that nonlinear functions change the expectation also affects other aggregates of MoD such as Maes' bitwise entropy $\overset{\text{Mb}}{H}$, which sums up the individual entropy of a Bernoulli RV per each identifier bit position, cf. (2.56). Entropy is usually associated with more is better, i.e. more secure, so it may be a quick reasoning to assume L bit of entropy as the optimum, which is equivalent to one bit of entropy per identifier bit position. However, one then falls victim to the same confusion of *maximum* vs. *optimum* as with Uniqueness. Even an ideal PUF will not produce 0.5 for each element in the MoD, but cause them to follow a binomial-like distribution with $p = 0.5$, D trials, and normalized support. So most of the elements of the MoD will be close to, but still different from, 0.5. An increase of D concentrates the values closer to 0.5, which makes the expectation of $\overset{\text{Mb}}{H}$ for an ideal PUF dependant on the number of analyzed devices as it is for Uniqueness according to Hori et al. This can again be verified from the PMF of the summands,

$$\mathrm{P}(M''' = m_l''') = \begin{cases} \binom{D}{D/2} 0.5^D & \text{if } m_l''' = 1, \\ \binom{D}{i} 0.5^{D-1} & \text{if } i \in \{0, 1, \dots, \lfloor \frac{D-1}{2} \rfloor\}, \\ 0 & \text{otherwise,} \end{cases} \tag{3.46}$$

where

$$\begin{aligned} i &= D m_l, \\ m_l''' &= -\left(m_l \log_2(m_l) + (1 - m_l) \log_2(1 - m_l)\right). \end{aligned} \tag{3.47}$$

Replacing Shannon entropy by min-entropy in this approach, now denoted $\overset{\text{b}}{H}_\infty$, does not change this behavior either, as visible from Fig. 3.7, where the expectations of various aggregates of MoD under a univariate Bernoulli model over its parameter p are plotted. The width of the rectangles, which show for each MoD aggregate the range of p that produces values within one standard deviation of the expectation for an ideal PUF, is identical for both $\overset{\text{Mb}}{H}$ and $\overset{\text{b}}{H}_\infty$ and only changes with D.

By inspection of the similarities of the recently discussed PMFs (3.42), (3.44), and (3.46), one may notice that the probability values remain the same and are only subject to the number of analyzed devices D, which is a direct result of the binomial-like distribution of MoD. So the change of expectation results solely from which value the aggregate takes for a particular MoD observation. To have the expectation independent of the number of analyzed devices, this mapping thus has to exactly compensate the dependence of probability values on D.

	$B = 1$	$B = 16$	$B = 100$	$B = 256$	$B = 1000$	$B = 4096$
$D = 5$	0.15811	0.03953	0.01581	0.00988	0.00500	0.00247
$D = 10$	0.07454	0.01863	0.00745	0.00466	0.00236	0.00116
$D = 15$	0.04880	0.01220	0.00488	0.00305	0.00154	0.00076
$D = 25$	0.02887	0.00722	0.00289	0.00180	0.00091	0.00045
$D = 50$	0.01429	0.00357	0.00143	0.00089	0.00045	0.00022
$D = 100$	0.00711	0.00178	0.00071	0.00044	0.00022	0.00011

Table 3.1.: Standard deviation of Uniqueness by Maiti et al. for an ideal PUF and selected values of D and B. For each D, the value for $B = 1$ is calculated numerically, while the remaining values are divided by $\sqrt{B}$.

3.6.3. Sensitivity

That Uniqueness according to Maiti et al. has a constant optimum of 0.5 in a range that starts from zero can lead to premature claims such as "we obtain the average inter-chip variation of 46.15%, which is pretty close to the ideal average of 50%" [32]. What is allowed to be called a "pretty close" value, however, *must* be related to the expected spread of the metric for an ideal PUF, which constitutes another contribution of this work. Common ways to represent spread are variance and standard deviation. Their exact meaning in terms of how probable it is that an observation further than say one standard deviation away from the expectation is encountered by chance depends on the particular PMF; and even differs between smaller and greater values for skewed distributions. However, they form the commonly agreed minimum effort on spread consideration to be done, in particular if the actual distribution is unknown.

Standard deviation of any aggregate of MoD expectably depends on D and KL respectively B, because they determine how many observations of the RV or RVs are taken into account. For ease of understanding, the univariate Bernoulli model is assumed in the following. From basic probability theory it is known that the variance of the mean among N IID RVs is σ^2/N if σ^2 is the variance of a single RV. To calculate the variance of either form of Uniqueness, it is therefore sufficient to divide the variance of a summand, which is calculated numerically due to the nonlinear effect of D, by KL respectively B. To exemplify this, the resulting standard deviation of Uniqueness according to Maiti et al. for several values of D and B is tabulated in Tbl. 3.1. For other aggregates such as bitwise entropy, which use a sum instead of a mean, the variance of their sum increases linearly with the number of observations. So a sum of N IID RVs with expectation μ, or N independent observations of the same RV, has expectation $N\mu$ and standard deviation $\sqrt{N}\sigma$.

While the standard deviations in Tbl. 3.1 seem rather small, which might be interpreted in a sense that Uniqueness would be a sharp metric for the ideality of the PUF, Fig. 3.7 shows a different picture: Because Uniqueness is the mean of a quadratic function of MoD, it changes least around $p = 0.5$, which is the most interesting region, though. Hence a Uniqueness result of e.g. 0.48 is the expectation for a severely biased PUF design with $p = 0.4$, but at the same time just about one standard deviation away from the expectation for an ideal PUF, considering a single-challenge PUF with $B = 16$, tested on $D = 10$ devices. In Fig. 3.7, this is emphasized by rectangles that are centered on the expectation for an ideal PUF, extending in the vertical axis one standard deviation in each direction, and are so wide that their bottom corners lie on the graph of expectation. The width of the rectangle is thus a measure of how sensitive the metric is to bias compared to the expected spread for an ideal PUF.

Unsurprisingly, this width is independent of which aggregate of MoD is used. This is again a result of the probability values being identical to all aggregates, which means that a change in expectation and standard deviation only comes from different aggregate values being mapped onto the outcomes of the MoD. So if an aggregate is to vary stronger for a change in p, its standard deviation will increase accordingly, because the expected variation of MoD for an ideal PUF is amplified in just the same way. An improvement of the informative value of one of these aggregates thus requires to increase the number of analyzed devices, or bit positions. An alternative way that may provide more insight without additional devices is to check for the distribution of MoD, which is a novel test in the field of PUFs and proposed in Sec. 4.1.2.

3.7. Confidence Intervals and Accuracy of Results

This section generalizes the question of sensitivity vs. the statistical spread to be expected for an ideal PUF, which was discussed for Uniqueness at the end of the previous section. Although the appropriate handling of significant digits and a discussion of achieved accuracy are elementary tools in every scientific field, few publications on PUFs discuss the issue at all. Among all the publications cited in this thesis, the question of accuracy is at least addressed in [67, 70, 72, 81] and CIs are provided in [67, 72, 81].

A first example of the consequences that can arise from a lack of accuracy consideration was already given at the beginning of Sec. 3.6.3: A Uniqueness of 46.15% was claimed to be “pretty close” [32] to the ideal 0.5, as if such a deviation where to be expected even for an ideal PUF. In this case, where $B = 128$ and $D = 15$, the standard deviation of Uniqueness for an ideal PUF is ≈ 0.0043, though. So the observed result differs significantly from the expectation for an ideal PUF.

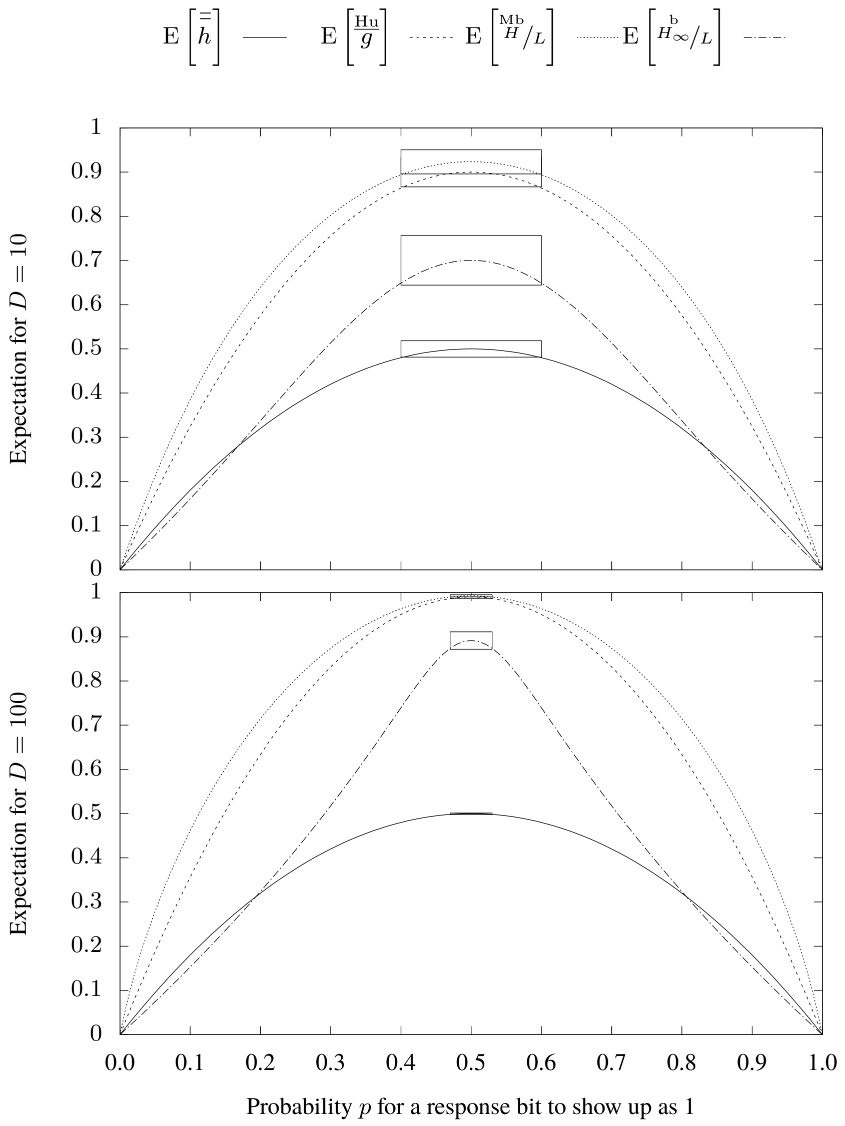


Figure 3.7.: Expectation of several aggregates of MoD under univariate Bernoulli model over parameter p. Rectangles mark the range of one standard deviation for $B = 16$ around the optimum.

An example for qualitative accuracy consideration was given by Maiti et al. [70], who showed the benefit of more sample devices using a sliding window method. For this, the "global average frequency" [70] of their ROs, presumably the mean among all accesses from all ROs on a device, within "groups of 16 FGPAs (1 to 16, 2 to 17 and so on)" [70] was plotted against the same average among all 125 FPGAs. This revealed that the averages of the groups varied between -3.5% and +3.4% compared to the average of the entire dataset. However, this does not yield quantitative information on how accurate the results for the various metrics reported in [70] are and no further accuracy consideration is done.

To use the standard deviation of a metric for an *ideal* PUF as accuracy hint for an *empirical* result of the metric is not necessarily sound, though. A valid statement would for example be that an observed result of 0.46 for Uniqueness according to Maiti et al. based on five devices with 16 bit response is consistent with the assumption that the PUF candidate performs well, because it is just about one standard deviation away from the expectation for an ideal PUF, cf. Tbl. 3.1. However, it would not be valid to state that this version of Uniqueness can always be measured with the standard deviations found in Tbl. 3.1, because they are based on the behavior of an ideal PUF. Standard deviation, however, depends on the parameters of the underlying probability distribution, which for Uniqueness are the Bit-Alias, i.e. the probability to observe a 1 response at a particular response bit position. So if a PUF candidate with 16 bit response would produce, say, a Uniqueness of 0.3 based on five sample devices, it is very probable that the actual probability distribution is not that of an ideal PUF and hence the standard deviation differs from the values in Tbl. 3.1.

To correctly represent the accuracy of a metric result therefore requires to consider the inherent variation of the statistical experiment for the given parameters and observations. One way to perform this is to use interval estimates rather than point estimates. The remainder of this section will commence with an explanation of the difference between them, followed by an example of how to obtain CIs for the binomial distribution, which applies to several existing metrics. This at hand, Sec. 3.7.3 discusses the implications for performance testing of PUF candidates regarding dataset size to achieve a given accuracy. A preliminary version of this discussion and the CI examples appeared in [**91**]. The section is then completed by an evaluation of CIs provided in [67, 72, 81].

3.7.1. Point Estimates vs. Interval Estimates

Estimation of unknown population parameters from sample data is one of the most basic tasks in statistics. In the PUF field, it occurs on numerous places, e.g. when the global bias of a PUF candidate, so the success parameter p under the univariate Bernoulli model, is estimated from the overall ratio of 1s and 0s in the dataset.

While point estimation is in general easier than interval estimation, it likely leads to incorrect conclusions if performed incautiously. This is best visualized in a trivial example: Let t be the number of bits equal to 1 in a dataset of N bits in total, produced by a PUF for which the IID assumption holds, then $t \leftarrow T \sim \mathfrak{B}(N, p)$. The true success probability p will remain unknown, but an estimation $\hat{p}$ can be made. The typical UMVU estimator for p is $\hat{p} = {}^{t}/_{N}$. So if one observes, say $t = 30$ for $N = 100$, the point estimate would be $\hat{p} = 0.3$. However, this estimate is most probably *incorrect*, except for $p = 0$ and $p = 1$, in which case $t = 0$ or $t = N$. Setting the question of *how* incorrect the estimate is, i.e. its distance from the true value p, aside for a moment and only considering the question whether the estimate *is* correct, this happens if and only if $t = pN$. Given that t and N are integer, this can hold only for $p \in \{0, \frac{1}{N}, \ldots, 1\}$ rather than $p \in [0, 1]$. So there remains only a countable set of $N + 1$ values for p where the point estimate $\hat{p}$ could possibly be correct. Now for the estimate to be actually correct for a given p and N, there is exactly one value for t among the $N + 1$ outcomes that makes the estimate correct, while the other N values lead to an incorrect estimate. Although the latter is the mode of the binomial distribution, i.e. the most probable outcome, the probability for the event that *any other* outcome occurs is at least as high and in most practical cases even much higher. Fig. 3.8 displays this for some exemplary N, p, where points indicate all p that allow $\hat{p} = p$ for a given N and the probability for it. The case $N = 2, p = 0.5$ is the only one where the probability for a correct estimate equals that for an incorrect estimate, because all four outcomes 00, 01, 10, 11 are equiprobable and 01, 10 both lead to the correct estimate. In all other cases, the probability for a correct estimate is lower than 0.5, i.e. it is more probable to observe an incorrect estimate than a correct one. For example, with $N = 20$, $p = 0.2$, the probability to observe the correct estimate is only 0.22, i.e. there is a 78% chance to observe an incorrect estimate. Note that these calculations are based on the assumption that the true distribution, which remains as unknown as its true parameters, matches the assumed binomial distribution.

For continuous RVs, such as the mean frequency of a set of ROs or some normal distributed RV $T' \sim \mathfrak{N}(\mu, \sigma^2)$ in general, the question whether the point estimate matches exactly with the expectation apparently becomes a paradox, because the probability to observe exactly μ in a single sample – or a set of samples that result in exactly μ as their estimation – is zero by definition. In general, the probability that a continuous RV turns out exactly as *any* particular value is zero, which is the reason why the derivative of the CDF for continuous RVs is a PDFs instead of PMFs. For normal distributed RVs and many other continuous RVs, the typical UMVU estimator for μ is the mean among all samples. In contrast to the binomial distribution, though, an increase in N does not change the probability of this estimation to match exactly, because it remains at zero. However, an increase in N means more samples contribute to the mean, which allows variations from

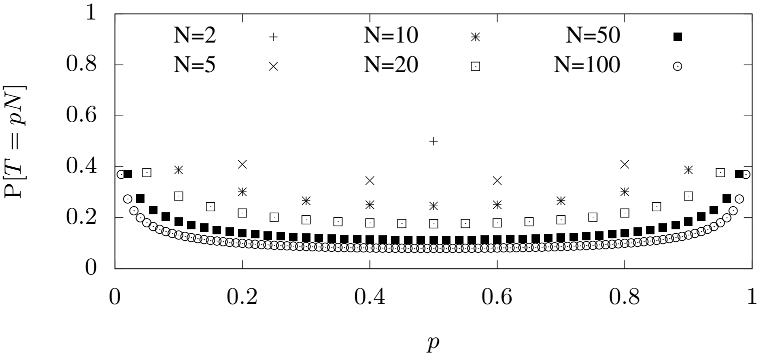


Figure 3.8.: Probability that the point estimate t/N equals the true success probability p of a $\mathfrak{B}(N, p)$ distributed RV T for some values of N, p.

individual samples to cancel each other out. Since the mean is an UMVU estimator, it will tend towards μ for $N \to \infty$ and do so with minimum variation for a given N for any μ. This leads to the question of *how* incorrect an estimation is.

A quick and practical, yet only approximate, way to answer this question is to assume the point estimate is correct and calculate the standard deviation under this assumption. Reusing the example from above, this means $\hat{p} = 0.3$ is assumed to match p and corresponds to a standard deviation of $\sigma = \sqrt{\hat{p}(1 - \hat{p})} \approx 0.458$ for the Bernoulli RV. Since the estimation in this example stems from $N = 100$ observations of the RV, the resulting standard deviation of the mean is reduced to $\sqrt{\frac{\hat{p}(1-\hat{p})}{N}} \approx 0.0458$. By the common *3-sigma rule* – which technically only holds for the normal distribution, but which is frequently applied as approximation in other cases by argument of universal normality – one could now say the experiment shows that p is likely somewhere between 0.25 and 0.35 or at most one sigma away, at least quite certainly between 0.20 and 0.40 or at most two sigma away, and the chance of p being less than 0.15 or above 0.45, so more than three sigma away, is negligible. If the estimation would be based on only 10 observations, the standard deviation of the mean would be 0.145, which weakens the statement to locate p likely within 0.15 and 0.45, at least quite certainly below 0.6, and have negligible probability only for values above 0.75. The issue with this approach is that it disregards the exact shape of probability distribution, where the error increases the more the actual probability distribution differs from a normal distribution. In the last example with $N = 10$, the difference in shape is large enough that the 3-sigma rule produces invalid results, as it would not rule out the hypothesis $p = 0$ for $t = 3, N = 10$, although a $\mathfrak{B}(10, 0)$ distributed RV would never produce any success and therefore contradict with the observation $t = 3$.

A more formal approach to specify the achieved accuracy of estimation from an experiment, which addresses the issues of the 3-sigma rule, is to calculate so-called CIs. The concept is to avoid the report of a single value $\hat{p}$ that is likely to be incorrect anyway, but instead provide only a range $[\hat{p}_l, \hat{p}_u]$ that includes the – fixed but unknown – true value p with high probability based on the assumed underlying probability distribution. CIs differ from the 3-sigma rule in two ways: First, the assumed underlying probability distribution is not necessarily normal. Second, the point estimate is not assumed to be the correct distribution parameter, but an event of non-negligible probability. The latter is an important difference, because the shape of the distribution often depends on its parameters. In contrast to the 3-sigma rule, where the shape is assumed to be fixed by the point estimate and a range of probable outcomes is given, CIs report a range of distribution parameters that make the observation an event of sufficiently high probability.

To quantify the notion of high probability, CIs use a confidence level $1 - \alpha$ similar to hypothesis tests, where common values are 0.95 or 0.99. In mathematical notation: For some RV T that follows an arbitrary distribution $\mathfrak{A}(\underline{\theta})$, an individual CI for parameter θ_0 fulfills

$$\mathrm{P}\left(l(T) < \theta_0 < u(T)\right) = 1 - \alpha \quad \forall \underline{\theta}, \tag{3.48}$$

where $l(\cdot)$ and $u(\cdot)$ are functions defined prior to any experiment but specific only to $\mathfrak{A}$. The remaining parameters in $\underline{\theta}$ need to be considered by the CI though they are not of immediate interest. A common example for this is the Student's t distribution for the CI for the mean of a normal distributed RV independent of its standard deviation. One may also construct *joint* CIs for multiple parameters at once, but this is not considered in this work because the binomial distribution, which appears most often in the field of PUFs, has only one parameter. For discrete RVs, the above formula may not have an exact solution, therefore *approximate* CIs relax it to

$$\mathrm{P}\left(l(T) < \theta_0 < u(T)\right) \approx 1 - \alpha \quad \forall \underline{\theta}, \tag{3.49}$$

while *conservative* CIs follow

$$\mathrm{P}\left(l(T) < \theta_0 < u(T)\right) \geq 1 - \alpha \quad \forall \underline{\theta}. \tag{3.50}$$

Construction of $l(\cdot)$ and $u(\cdot)$ is not trivial, since it requires to invert a generalization of the probability distribution, i.e. to find a suitable quantile distribution. A common approach is to invert statistical hypothesis tests, which is also the way the CIs in the following subsection have been constructed [92]. The choice of which hypothesis test to invert or which other method of construction to use can result in multiple different approaches being common even for the same type of distribution. So to verify a given CI, it is helpful to know which approach has been followed.

3.7.2. Evaluation of Binomial Proportion CIs for PUF Testing

The binomial distribution is chosen to provide an example of how to use CIs in the field of PUFs, because it applies to many existing metrics, which operate on binary response data and calculate the mean or sum of several response bits under an IID assumption. The binomial proportion CI itself is a well explored problem and manifold methods can be found in textbooks. Among them, the following three constitute an adequate preselection, because they range from quite simple to rather sophisticated and two of them turn out useful in the PUF field for individual purposes. First, the so-called Wald CI

$$\hat{p}_{l,u} = \hat{p} \pm z\sqrt{\frac{\hat{p}(1-\hat{p})}{N}}, \tag{3.51}$$

which is constructed as the inversion of a Wald test [92]. Second, Wilson's score interval

$$\hat{p}_{l,u} = \frac{\hat{p} + \frac{z^2}{2N}}{1 + \frac{z^2}{N}} \pm \frac{z}{1 + \frac{z^2}{N}}\sqrt{\frac{\hat{p}(1-\hat{p})}{N} + \frac{z^2}{4N^2}}, \tag{3.52}$$

which is constructed as the inversion of a score test [92]. Both use the point estimate $\hat{p} = \frac{t}{N}$ and

$$z = \mathfrak{N}^{-1}\left(1 - \frac{\alpha}{2}, 0, 1\right), \tag{3.53}$$

i.e. the $1 - \frac{\alpha}{2}$ quantile of a standard normal distribution. Third, the so-called *exact* interval by Clopper and Pearson

$$\begin{aligned} \hat{p}_l &= \begin{cases} \mathfrak{Beta}^{-1}\left(\frac{\alpha}{2}, t, N-t+1\right) & t > 0 \\ 0 & \text{otherwise} \end{cases} \\ \hat{p}_u &= \begin{cases} \mathfrak{Beta}^{-1}\left(1-\frac{\alpha}{2}, t+1, N-t\right) & t < N \\ 1 & \text{otherwise} \end{cases} \end{aligned}, \tag{3.54}$$

which is constructed as the solution to an equal tailed binomial test [92]. Fig. 3.9 depicts the point estimate and CIs according to all three methods for $\alpha = 0.01$ and $N \in \{10, 100\}$. It shows that the difference between them is strongest for small N and $\hat{p}$ close to its axiomatic limits. In this area, the Wald method is not applicable, as it produces interval limits outside $[0, 1]$, which fail at the fundamental axioms of probability theory.

Since the binomial proportion CI is a well explored problem, evaluations of and comparisons between many approaches are available, e.g. by Agresti and Coull [92]. They found the Wald method to perform poorly for small N, providing either too wide CIs for $\hat{p} \approx 0.5$ or far too narrow CIs for $\hat{p}$ close to $\{0, 1\}$. Since the

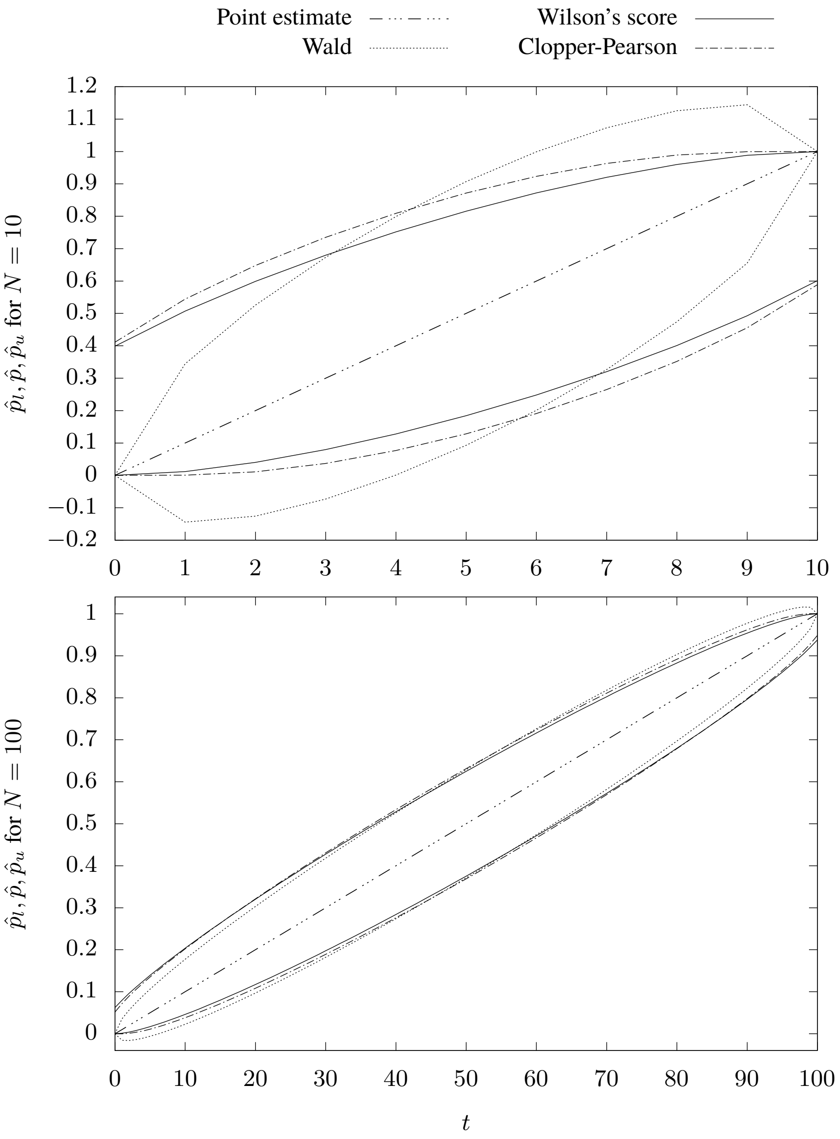


Figure 3.9.: Comparison of three methods to calculate the binomial proportion CI for $\alpha = 0.01$ and respective point estimate.

Wald method matches the CI for a normal distributed RV with standard deviation estimated to $\hat{p}(1-\hat{p})$, this fits the common rule of thumb that restricts the normal approximation of a binomial distribution to large N with a sufficient number of successes *and* failures. The Clopper and Pearson method is found in [92] to provide slightly too wide CIs, because it follows a conservative approach, cf. (3.50). This means it ensures at least $1-\alpha$ coverage probability even at worst case values of N, $\hat{p}$, which requires a lot of overhead since the binomial distribution has discrete support. In conclusion, [92] recommends Wilson's score interval, because its mean coverage probability is closest to – though not necessarily above – the desired level $1-\alpha$ for nearly all values of N, $\hat{p}$. Wilson's score interval therefore seems a good choice to determine CIs after a PUF experiment, because it combines simplicity with sufficient precision even for small N. The latter is especially relevant if the IID assumption does not hold for a PUF candidate and the CI is calculated for the MoD instead, since the number of observations then equals D rather than BCD.

3.7.3. Implications of CI Consideration on PUF Testing

Focusing on the width of the CIs in Fig. 3.9 as a concrete measure for the accuracy achieved in a PUF candidate's evaluation, two main conclusions can be made. The first is that independent of the chosen approach a precise estimation of p is most difficult around $\hat{p}=0.5$, which is the region of interest when it comes to evaluation of unpredictability. By inspection of Fig. 3.10, which focuses on this issue, one may notice that the width $\hat{p}_\Delta$ of the CI using e.g. Wilson's score interval is $\hat{p}_\Delta=0.499$ for $\hat{p}=0.5$, but $\hat{p}_\Delta=0.249$ for $\hat{p}=0$, when $N=20$, $\alpha=0.01$. This effect becomes the stronger the smaller the CI is supposed to be, as illustrated in Fig. 3.11. To achieve $\hat{p}_\Delta=0.5$, 20 observations are required at $\hat{p}=0.5$, whereas 7 observations suffice at $\hat{p}=0$. For $\hat{p}_\Delta=0.1$, the difference grew already from 60 observations at $\hat{p}=0$ to over 600 observations at $\hat{p}=0.5$. So in comparison to reliability analysis that might be familiar from other fields of electronics design, which usually aims to estimate e.g. a failure rate close to 0 or the yield during wafer processing close to 1, unpredictability analysis requires many more observations to achieve the same CI width. This also means care must be taken when rules of thumb from reliability analysis shall be applied to unpredictability analysis.

The second conclusion is that the *resolution* of the point estimate, i.e. $^1/_N$ so e.g. 0.01 for an experiment with 100 observations, is far off the accuracy of estimation in terms of CI width, see Fig. 3.12. While resolution starts from 0.5 for $N=2$ and reaches 0.1 for $N=10$, CI width reaches 0.1 only with $N\approx 660$ by any of the three methods. To emphasize the importance of this conclusion, Fig. 3.13 depicts the achievable CI width for the MoD in selected previous work. Included are

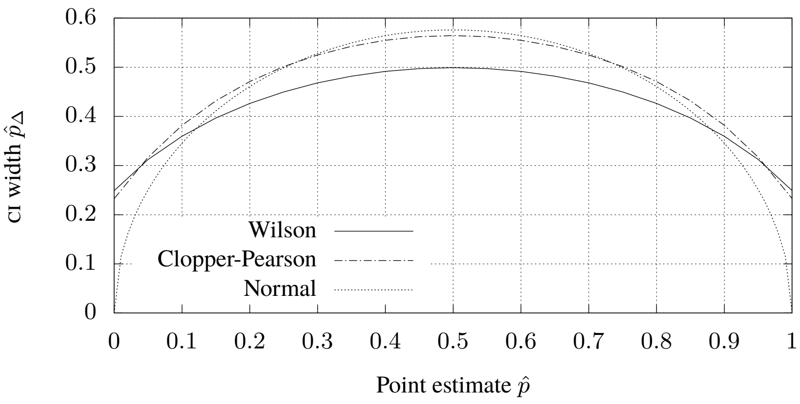


Figure 3.10.: Width of CI according to Wilson's score, Clopper-Pearson, and Wald over point estimate $\hat{p}$ for $\alpha = 0.01$, $N = 20$. A CI of 0.5 ± 0.1 has a width of 0.2. Reproduced from [**91**].

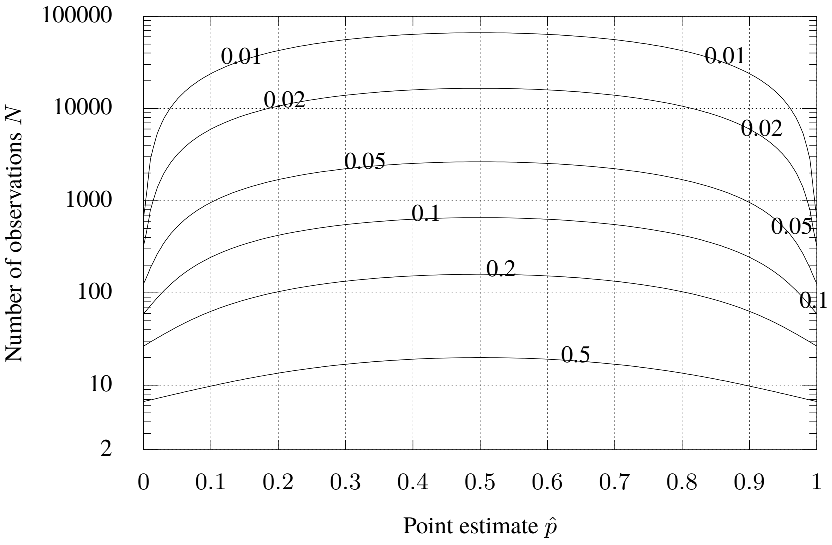


Figure 3.11.: Required number of observations to achieve given CI width $\hat{p}_\Delta$ using Wilson's score interval over point estimate $\hat{p}$ for $\alpha = 0.01$.

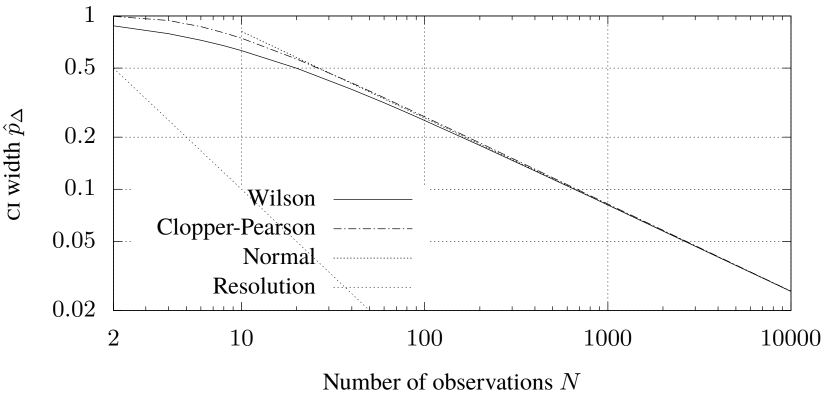


Figure 3.12.: Resolution of point estimate compared to width of CI according to Wilson's score, Clopper-Pearson, and Wald over number of observations N for $\alpha = 0.01, \hat{p} = 0.5$. A CI of 0.5 ± 0.1 has a width of 0.2. Extended version compared to [**91**].

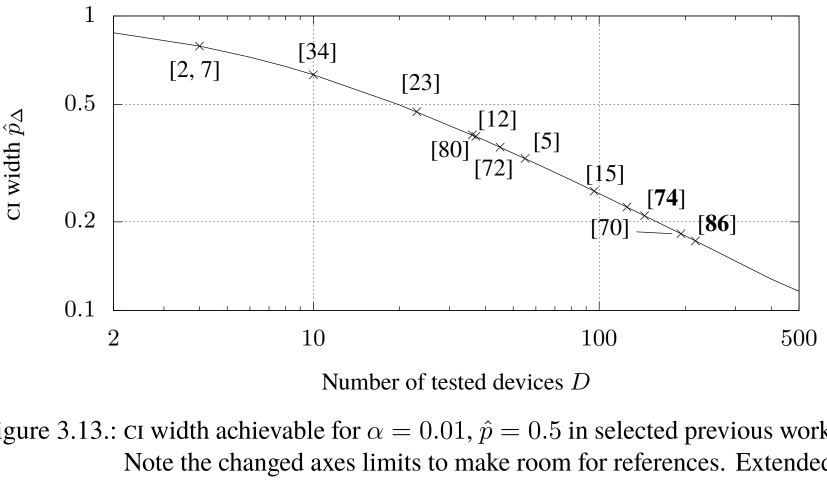


Figure 3.13.: CI width achievable for $\alpha = 0.01, \hat{p} = 0.5$ in selected previous work. Note the changed axes limits to make room for references. Extended version compared to [**91**].

publications between years 2000 and 2018 that analyzed an outstanding number of devices when they were published and – for context – some high impact publications. While [5] provided results based on 55 ASICs already in 2000, most publications use less than 20 devices, even high impact ones such as [2, 7], which analyzed four devices. Notable exceptions are [23] with 23, [12] with 37, [80] with 36, and [72] with 45 analyzed devices. In 2010, Maiti et al. [70] published data from 125 FPGAS and extended the dataset in 2011 to 193 devices, making it the first to allow Bit-Alias estimation better than plus or minus ten percentage points. Post 2010, noteworthy publications regarding number of analyzed devices are [15] with 96, [**74**] with 144, and [**86**] with 217 analyzed devices, being the first to exceed Maiti's 2011 record, while high impact publication [34] analyzed ten devices. Since the accuracy of MoD estimation also affects metrics such as Uniqueness or bitwise entropy, cf. Sec. 3.6, this finding severely impugns performance claims in most publications, in particular those not listed in Fig. 3.13, because these claims are based on too little data.

To ensure the intended accuracy can be achieved, a quick approximation of the required number of observations to achieve a given CI width for unpredictability analysis can be made even from the simple Wald CI, because reasonable settings avoid its region of poor performance. Given, for example, the request to estimate p better than $\pm.05$ at a 0.99 confidence level, Fig. 3.12 shows that several hundred observations are required and the difference between the three discussed methods for CI calculation is negligible. The Wald CI may thus be rearranged into

$$N_{\hat{p}=0.5}(\hat{p}_\Delta, z) = \left(\frac{z}{\hat{p}_\Delta}\right)^2, \tag{3.55}$$

where $\hat{p} = 0.5$ has been fixed because CI width is largest at this point and this case has to be expected for unpredictability analysis. For a $1 - \alpha = 0.99$ confidence level, $z \approx 2.5759$, which yields 664 observations to achieve $\hat{p}_\Delta = 0.1$. A cross check using the Clopper-Pearson method results in 680 required observations, while the Wilson's score interval requires 658 observations for this CI width.

Note that all calculations and conclusions in this subsection expect the observations to be IID. If there are dependencies among the samples, the effective number of informative observations is reduced, so additional observations are required to compensate for it. If observations do not belong to the same distribution, e.g. because the PUF candidate circuit fits rather the multivariate Bernoulli model than the univariate model, individual CIs have to be calculated, of course.

3.7.4. Review of CIs Provided in Previous Work

There are few publications in the field of PUFs that provide CIs for their results, among them Su et al. [67], Hori et al. [72], and Kömürcü and Dündar [81].

This section aims to verify their work by recalculation of the CIs, which includes testing different methods because the way the CIs are found were not always fully explained. Furthermore, the section discusses the validity of the claimed CIs and possible improvements.

CIs in “A Digital 1.6 pJ/bit Chip Identification Circuit Using Process Variations”

Su et al. [67] provide CIs on a 0.95 confidence level for the mean inter-class HD, the mean along devices of the “number of unstable bits” [67], and the mean along devices of the number of 1s in the response. It is neither explicitly stated how these CIs are constructed, nor which distribution is assumed. However, given that a normal distribution is fitted to the respective experimental data in some of their figures and the CIs are always symmetric about the point estimate, the assumption is that they used the normal distribution also for their CIs. While this technically constitutes a modeling error already, since all three metrics have limited support, which contradicts with the normal distribution that has the entire set of real numbers as support, it might be an acceptable approximation under certain circumstances, cf. the Wald CI for the binomial proportion CI. Two questions will therefore be addressed in the following: First, whether the assumption that Su et al. used a normal distribution to infer CIs is true, and, second, how these approximation based CIs perform compared to exact CIs based on the univariate Bernoulli model.

As a prerequisite to address the first question, the textbook approach to construct a CI for μ of a normal distribution from N independent observations is recalled:

$$\hat{\mu}_{l,u} = \hat{\mu} \pm \mathfrak{T}^{-1}\left(1 - \frac{\alpha}{2}, N-1\right) \frac{s}{\sqrt{N}}, \tag{3.56}$$

where $\mathfrak{T}^{-1}\left(1 - \frac{\alpha}{2}, N-1\right)$ is the $1 - \frac{\alpha}{2}$ quantile of a Student’s t distribution with $N-1$ degrees of freedom and s is the empirical standard deviation among the observations t_i,

$$s = \sqrt{\frac{1}{N-1} \sum_{i=1}^{N} (t_i - m)^2}, \text{ with} \tag{3.57}$$

$$m = \frac{1}{N} \sum_{i=1}^{N} t_i. \tag{3.58}$$

Since this requires the empirical standard deviation, recalculation of the CIs is only possible due to the fact that [67] is one of the few commendable publications that contain raw data, more precisely the true response for all 19 devices.

Beginning now with the mean of the number of 1s per response, or mean Uniformity as referred to by Maiti et al., Su et al. provide a CI of 0.5016 ± 0.0198 for their common-centroid design and 0.5115 ± 0.0199 for their symmetric design [67]. Following the assumption of a normal distribution, this should be the CI to achieve for a confidence level of $1 - \alpha = 0.95$ from $N = 19$ observations, one per device, of RV $T \sim \mathfrak{N}(\mu, \sigma^2)$ that models the number of 1s in a response. Performing this for the data from the symmetric design confirms $m = 0.5115$, but provides $s = 0.0634$, which gives a 0.95 level CI for the mean of 0.5115 ± 0.0305. Su et al. must have used a different method.

Considering that the quantity in question is a proportion rather than a true normal distributed RV, the variance of the normal distribution my be approximated by $m(1-m)$ instead of the empirical variance of the observations. The CI formula then coincides with that of the Wald CI for the binomial proportion except that the latter uses the normal quantile distribution rather than the t quantile distribution, cf. Sec. 3.7.2. If now the univariate Bernoulli model is applied, every response bit constitutes an observation, rather than every device, so $N = DB = 19 \cdot 128 = 2432$. The Wald CI with $1 - \alpha = 0.95$ and this parameters yields 0.5115 ± 0.0199 and repeating this approach for the data from the common centroid design yields 0.5016 ± 0.0199. Although the CI for the common centroid design reported in [67] is 0.5016 ± 0.0198, this small difference in width may be due to rounding, so it can be concluded that what has been used by Su et al. for Uniformity in [67] is the Wald CI under a univariate Bernoulli model.

Based on the findings from Sec. 3.7.2, the Wald CI performs well for $N = 2432$ and $\hat{p} = 0.5115$ or $\hat{p} = 0.5016$, so its difference from the Wilson's score CI should be negligible. Verification with Wilson's method shows that the limits round to the same numerical values as the Wald method. So as long as the univariate Bernoulli model applies, the CIs given in [67] for the mean of Uniformity can be considered accurate.

Turning to the CIs for mean of the "number of unstable bits" [67], Su et al. state 3.89 ± 0.27 for the symmetric design and 4.84 ± 0.33 for the common centroid design, which equal a ratio of 0.0304 ± 0.0021 and 0.0378 ± 0.0026 with regard to the length of a device's response. To calculate CIs with any of the before mentioned approaches requires the assumption that all response bit positions on all devices have the same a priori probability of being unstable. A Wald CI with confidence level $1 - \alpha = 0.95$ and $N = 2432$ would produce ± 0.0068 for $\hat{p} = 0.0304$ and ± 0.0076 for $\hat{p} = 0.0378$. Since these CIs are wider than those reported in [67], one may conclude that multiple accesses to the PUF candidate circuit have been performed, increasing N. Because no information is given in [67] on how many accesses have been made, this information needs to be estimated from the width of the CIs. However, no common number of accesses can be found that would yield CIs that match those reported in [67] for both the symmetric and common centroid design. For the symmetric design, 11 accesses would be necessary to reduce the CI from ± 0.0068 to ± 0.0021, but for the common centroid design 8 respectively 9 accesses suffice to achieve a CI of ± 0.0027 respectively ± 0.0025. So either a different number of accesses has been made to the designs, or a different method has been used.

In contrast to the number of 1s in the response, the number of unstable bits has $\hat{p}$ close to zero, which is a region where the Wald method likely performs poor and the CI should no longer be symmetric around the point estimate. It therefore seems advisable to compare these CIs to those from e.g. the Wilson method. For $1 - \alpha = 0.95$, $N = 2432$, this provides 0.0243 to 0.0380 or $0.0304(-0.0061/+0.0076)$ for the symmetric design and 0.0309 to 0.0461 or $0.0378(-0.0069/+0.0083)$ for the common centroid design. The effect of $\hat{p}$ being close to zero is partially compensated by the large value of N, so the widths of the CIs are almost the same between the Wald and the Wilson method. However, the Wilson method represents the asymmetry of the CI with regard to the point estimate, the Wald method does not. This can be relevant, because to ignore the asymmetry means to underestimate the upper limit of the ratio of unstable bits to be expected.

To conclude the analysis of the "number of unstable bits" [67], the CIs cannot be reproduced without further information. Although the Wilson method would fit better to the scenario, the CIs are symmetric, which suggests that Su et al. used the Wald method. With the Wald method, the CIs can only be reproduced under the assumption of a different number of accesses per layout, though.

Verification of the CIs for the mean inter-class HD, or Uniqueness as referred to by Maiti et al., starts with the interesting question of how many observations have been made. Based on the provided CIs, Su et al. used the textbook approach for normal distributed RVs in (3.56) with $N = 171$ and s from (3.57). This produces 0.5055 ± 0.0066 for the mean FHD of the symmetric design and 0.5013 ± 0.0063 for the mean FHD of the common centroid design, which scale to 64.70 ± 0.85 and 64.16 ± 0.80 in HD and match the CIs 64.70 ± 0.84 and 64.16 ± 0.80 given in [67] up to a small rounding error. However, while the mean is based on all $D(D-1)/2 = 171$ possible pairs of devices, it is obvious that they do not constitute 171 independent observations, because every device participates in 18 of these HDs. A conservative approach that assumes $N = D$ independent observations from a normal distribution would yield a $1 - \alpha = 0.95$ CI of $0.5055(\pm 0.0211)$ FHD, or $64.70(\pm 2.70)$ HD, for the symmetric design and $0.5013(\pm 0.0200)$ FHD or $64.16(\pm 2.56)$ HD for the common centroid design.

However, based on the PMF of the mean inter-class HD developed in Sec. 3.6, a normal approximation of the mean inter-class HD seems oversimplified as it ignores the PMF's skewness. Considering furthermore that mean inter-class HD is just an aggregate of the MoD, it seems more reasonable to obtain a CI for the MoD instead. Under the univariate Bernoulli model, though, this boils down to the very same CI as for Uniformity, which is already discussed above.

CIs in "Quantitative and statistical performance evaluation of arbiter physical unclonable functions on FPGAs"

Hori et al. [72] use the canonical normal distribution approach in (3.56) for all CIs and thereby perform the same oversimplification as Su et al. [67], because their metrics, again, have a very small range of support. The intended support according to [72] is the interval $[0, 1]$, though Correctness is found to have support $[-1, 1]$, cf. Sec. 3.3.4, and Uniqueness for individual devices from (2.28) has support $[0, 2]$, cf. Sec. 2.3 and [**74**]. The metrics produce one result per device, which are interpreted as D independent observations, and m, s are estimated from them. To verify whether the assumption of normality happens to be justified despite the limited support, an Anderson-Darling (AD) test for normality can be performed, since Hori et al. published their raw data.

However, the results printed in Table II of [72], which contains mean and standard deviation among devices for all metrics, cannot be reproduced even with the original code by Hori et al. available for download together with their raw data from [73]. Tbl. 3.2 contrasts the originally published values from [72] with those obtained through data and scripts from [73]. The errors for Randomness, a device's overall probability for a 1, and Diffuseness seem small enough to arise from improper rounding, but for Steadiness and Correctness the errors appear too large for this explanation. A reimplementation of the metrics by Hori et al. in Python leads to the same results as produced by their analysis scripts. This suggests that either the dataset available for download is different from the dataset used during preparation of [72], or the analysis scripts have been changed in the meantime, or both.

An AD test on the metric results obtained through the reimplementation, which also includes Uniqueness, fails at the $\alpha = 0.01$ significance level for four out of six metrics. For Randomness and a device's overall probability for a 1, the observed distributions are too skewed to be normal. For Diffuseness and Uniqueness, the outliers are too strong. Histograms of all metrics can be found in Fig. 3.14. Although a normal approximation would be justified for Steadiness and Correctness with this particular dataset, those metrics are also the ones where the difference between the values published by Hori et al. and those found by reimplementation is largest. As the AD test fails, the claim of normality by Hori et al. for all of their metrics therefore has to be doubted, which also renders the CIs provided by Hori et al. invalid, since they are based on the assumption of normality. However, the following suggestions for improvement can be made:

Randomness and Steadiness contain logarithms, which Hori et al. correctly

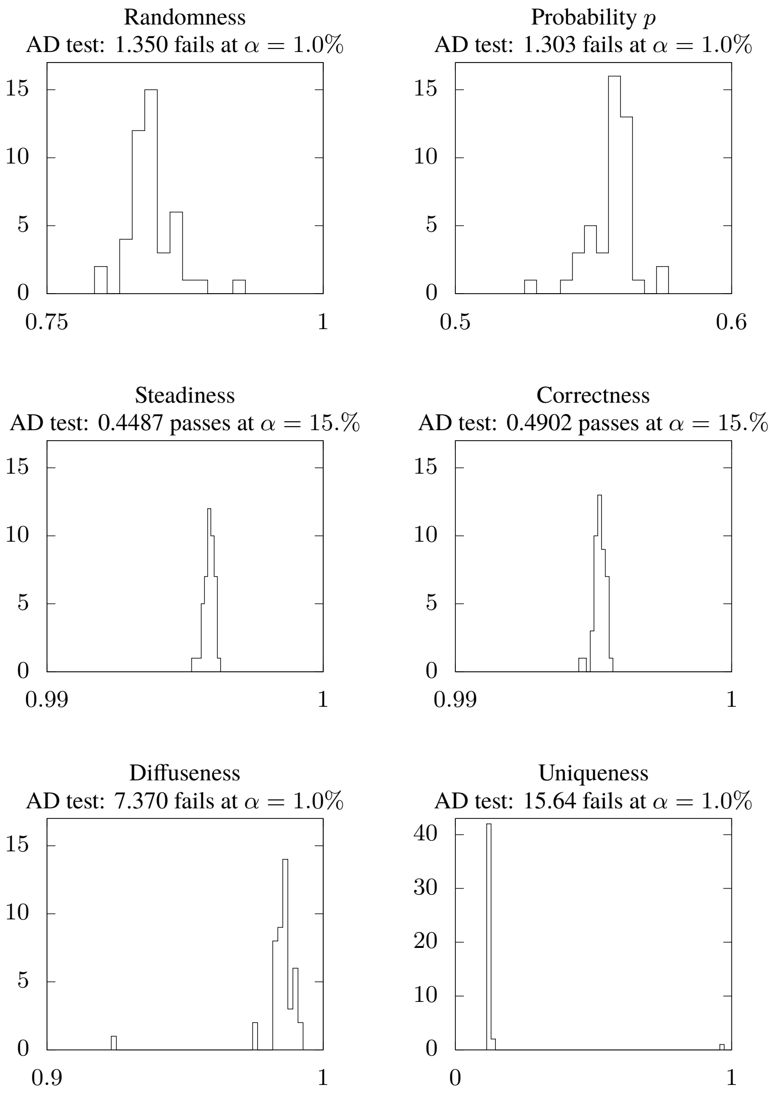


Figure 3.14.: Distribution of results of metrics according to Hori et al. [72] for data published by Hori et al. [73]. For Uniqueness the per-device version (2.28) has been used, so all six histograms are based on $D = 45$ observations.

	Printed in [72]		Recalculated via [73]	
	m	s	m	s
Randomness	0.8469	0.02670	0.8447	0.02283
Probability for a 1	0.5561	0.01017	0.5569	0.00875
Steadiness	0.9848	0.07401	0.9959	0.00019
Correctness	0.9829	0.08293	0.9952	0.00022
Diffuseness	0.9839	0.01021	0.9843	0.00991
Uniqueness	0.3675	0.5150	–	–

Table 3.2.: Comparison of results printed in [72] to results obtained by use of the originally published analysis scripts on the originally published dataset, both available for download from [73]. The analysis scripts published by Hori et al. in [73] do not contain code to calculate Uniqueness.

identify to hinder a normal approximation. For Randomness, they therefore calculate the CI on the argument to the logarithm instead and take the logarithm of the CI limits, which utilizes the *invariance* property of CIs. This means the CI is calculated for the proportion of response bits in all identifiers from all devices, or p of the univariate Bernoulli model. For Steadiness, they do not follow this approach, however the argument to the logarithm is the proportion of accesses that produced the correct response bit, which is the prime example for a binomial distribution rather than a normal distribution. So instead of a normal approximation with standard deviation estimated from the samples, one of the existing methods to calculate a CI for the binomial proportion, such as those from Sec. 3.7.2, could have been used to obtain more appropriate CIs for the arguments of the logarithms.

For the metrics Correctness, Diffuseness, and Uniqueness, the issue is again the assumption that every possible pair along some axis produces entirely independent samples. As discussed previously, this is a strong assumption that carries the risk of overestimating the confidence in the unpredictability or reliability of a PUF candidate. Furthermore, they are all aggregates of the mean along some axis, which suggests to calculate a CI for those means instead.

CIs in “Determining the quality metrics for PUFs and performance evaluation of Two RO-PUFs”

Kömürcü and Dündar [81] attempt to use Chebyshev’s theorem to provide CIs for their metrics. Chebyshev’s theorem states that for an observation t of a RV T, which has an arbitrary distribution with known expectation μ and standard deviation σ,

$$\mathrm{P}\left(|T-\mu| \geq k\sigma\right) \leq \frac{1}{k^2}. \tag{3.59}$$

Its advantage is that it holds for any probability distribution with known finite expectation and variance, so it does not require to determine the kind of distribution as e.g. normal or binomial. The price for the more general applicability are much wider CIs. For example, it permits to claim that an observation will be within *ten* standard deviations from the distribution's expectation with probability 0.99. If the distribution would be known to be normal instead, the 3-sigma rule would allow to claim that an observation is within *three* standard deviations with probability 0.997.

The idea to use Chebyshev's theorem instead of an assumption of normality is to be welcomed, since it removes the burden to justify this assumption. However, the way it is used in [81] appears to be incorrect or at least insufficiently explained. This makes it infeasible to verify their approach.

3.8. Unpredictability Evaluation by Compression of Responses

The findings presented in this section are based on the results of the undergraduate internship of Martin Radev under my supervision [93].

3.8.1. Optimality of CTW for Entropy Estimation

Since Ignatenko et al. [77] claimed that CTW is optimal – in the sense that it approaches the source entropy – to estimate the secrecy-rate of two-dimensional data from fuzzy sources such as biometrics and PUFs, it has also been used to compress responses of PUF candidates to obtain an estimate of their entropy, e.g. in [15, **71**, 76]. The idea behind this approach is that if the algorithm is able to approach the source entropy, it will automatically detect and take into account dependencies between the response bits and therefore provide a tighter bound than e.g. bitwise entropy calculated from MoD, which only considers the probability for a 1 or 0 at each response bit position. However, the conditions under which the algorithm approaches the source entropy have to be considered, as they might not be met during the evaluation of some PUF candidate. Two examples support this consideration: First, bitwise entropy of the ROmaiti dataset with bits created by MEPWC was found to be $\approx 0.94\,\mathrm{bit/bit}$, but CTW achieved a mere $\approx 0.98\,\mathrm{bit/bit}$ entropy bound [**71**]. Second, modern file compressors such as PAQ [94] and CMIX [95] are able to achieve higher compression rates, thus tighter bounds on entropy, than CTW for the SRAMxmc dataset [93]. The latter example will be further discussed in the following.

Lossless compression can provide an upper bound on the entropy of a sequence, because it removes redundancy within the sequence. Historically, such as with the widespread DEFLATE algorithm [96], this was achieved mainly by replacement of repeated blocks through back references, which for the case of DEFLATE are then encoded together with the literal blocks through multiple Huffman trees. This provides a practical trade off between (de-)compression speed and compression ratio. For the intended application, though, compression speed is of subordinate priority, because it does not affect the entropy estimation. Furthermore, decompression speed is entirely unimportant as the compressed data is never decompressed, but only compared in size to the original data to estimate compression ratio. This allows to focus on so-called statistical compression techniques, which include CTW, because they offer the best compression ratio at the price of a comparatively high demand in resources. Instead of a mere search for repeated blocks, statistical compressors use an information theoretical source model, whose parameters are adjusted as the sequence is processed a bit at a time and which provides a prediction for the next bit in the sequence. The required amount of storage for the model parameters and exceptions determines the size of the compressed data and thus compression ratio, which is an upper bound on the entropy of the input sequence. The tightness of this bound depends on whether the model is adequate for the sequence and its parameters and exceptions can be represented efficiently.

The statistical compressors compared herein are:

CTW [97], which was invented by Willems et al. in 1992 and uses a tree source with memory, where the next bit is the result of a Bernoulli distribution whose probability depends on the outcomes of a given number of previous bits. The version used for this work is the reference implementation v0.1 that used to be available from TU Eindhoven's website and was executed with default options.

PAQ [94], which is a series of compressors founded by Matt Mahoney in 2002 and uses a concept called *context mixing*, which means to use a collection of different source models and combine their predictions, in this case by means of a neural network, to automatically favor the best source model for a wide variety of sequences. The version used for this work is PAQ8PX v182 with compression level 9 (highest).

CMIX [95], which is another series of compressors by Byron Knoll that also uses the context mixing concept and is partially based on PAQ8. The version used for this work is v18.

Both PAQ and CMIX achieve very good results and severely outperform CTW in open benchmarks such as the "Large Text Compression Benchmark" [98], which suggests they may outperform CTW in compression of PUF responses, too, and thus provide tighter bounds on entropy.

Board ID:	2A	06	4A		89	7D	27
CTW	0.3086	0.3596	0.3791	...	1.0013	1.0013	1.0013
PAQ	0.2703	0.3134	0.3297	...	0.9998	0.9999	0.9999
CMIX	0.2688	0.3117	0.3281	...	0.9996	0.9996	0.9996

Table 3.3.: Compressed size divided by original size of true responses from SRAMxmc dataset with CTW, PAQ, and CMIX statistical compressors in bit/bit sorted by ratio for PAQ.

To test this hypothesis, the true responses of the SRAMxmc dataset, as one binary file of 160 KiB size per device, were compressed one file at a time with each of the three compressors. Both CMIX and PAQ outperform CTW for every file and CMIX slightly outperforms PAQ for the majority of files. Tbl. 3.3 shows the achieved compression ratio for all three compressors and the three most and the three least compressible devices.

3.8.2. Statistical Modeling of PUF Outputs by Compressibility

That modern statistical compressors based on the context mixing concept outperform CTW, which uses only a tree source model, suggests that PUFs – or at least the SRAM PUF used here – are more accurately modeled by a different type of information theoretical source. To identify this type of source, modification and analysis of the inner workings of the context mixing stage of the compressors was necessary. Despite the slightly better performance of CMIX, the following experiment was performed with PAQ, since it has more documentation available. For this part, PAQ was run with compression level 4.

PAQ uses a neural network for the context mixing stage. So an obvious approach to identify the model that fits best to the PUF candidate would have been to inspect the partial derivatives of the weights given to each model as the network processes the data. However, this would have produced a large amount of data since there are approximately 1000 model-context combinations and 163 840 bytes to process [93]. While this approach would have allowed to observe how the neural network learns which model fits best to the currently processed section of the file, it would not have directly answered the question of which model fits best to the file as a whole. A more practical approach instead was to disable one or all but one model and compare the compressed size. This automatically yielded the type of source that models the entire response of the PUF candidate best and avoided the need to analyze a large number of partial derivatives, which change on every bit of input. The model that has the highest impact on compressed size when included or excluded is expected to model the PUF candidate best.

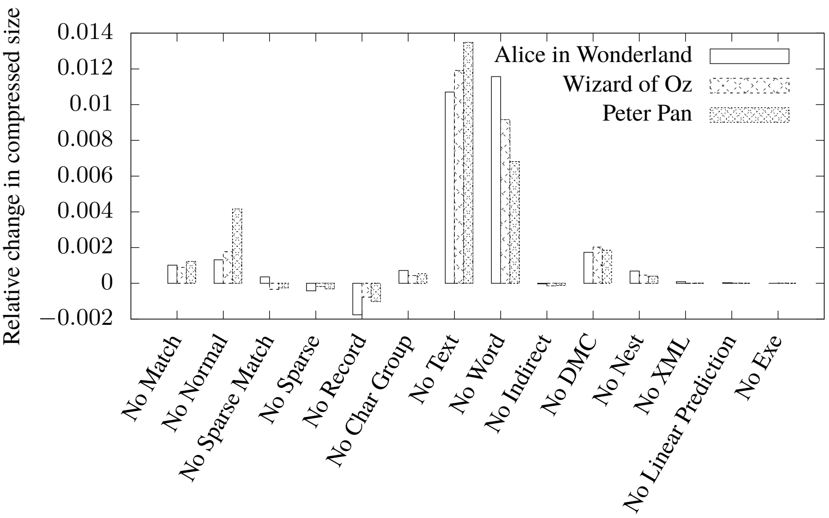


Figure 3.15.: Change in compressed size with individual models disabled relative to compressed size with all models in PAQ enabled for common text compression examples. Lower is better. Absolute compressed size with all models enabled: Alice in Wonderland 33 438 B, Wizard of Oz 50 348 B, Peter Pan 102 541 B Absolute uncompressed size: Alice in Wonderland 148 574 B, Wizard of Oz 222 318 B, Peter Pan 439 332 B

The latter approach was verified on three common text examples with the change in compressed size relative to that with all models enabled, which is shown in Fig. 3.15. As expected, to disable the text or word model has the worst effect on compressed size. Removal of the record model improves compressed size the most, which may indicate that for this model the neural network requires the most data to learn its unfitness and lower the weight for its predictions in favor of more fit models. XML, linear prediction, and exe model have the least influence on compressed size, so their predictions are quickly ignored by the neural network.

Fig. 3.17 shows the same approach applied to the true responses of the SRAMxmc dataset. Removal of the text or word model provides the most improvement for all devices, which at first glance seems obvious because the response data does not contain any text, but in fact means these models are the most difficult to detect as unfit for the neural network. For the high entropy devices, see Fig. 3.17b, removal of the match model has the worst impact on compressed size, closely followed

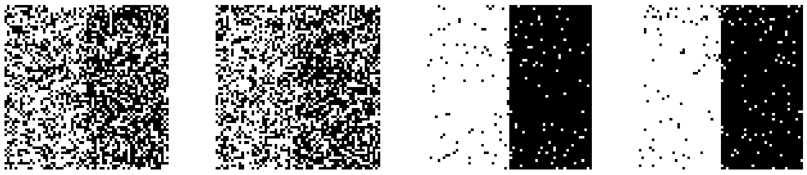
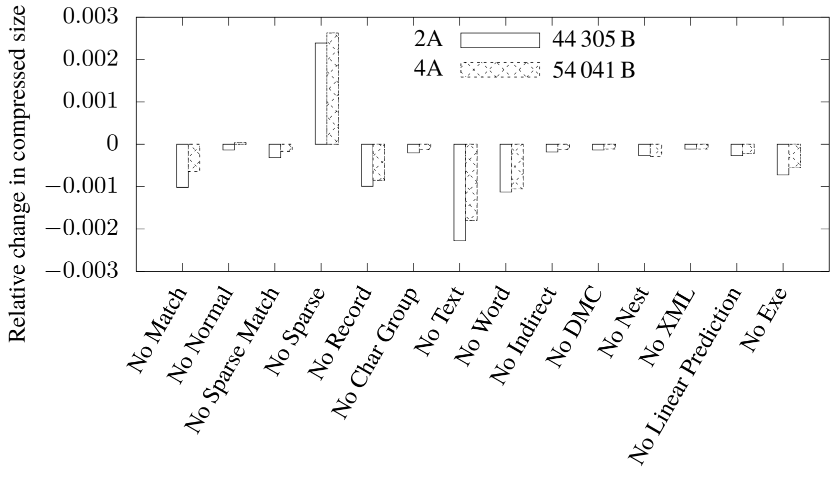


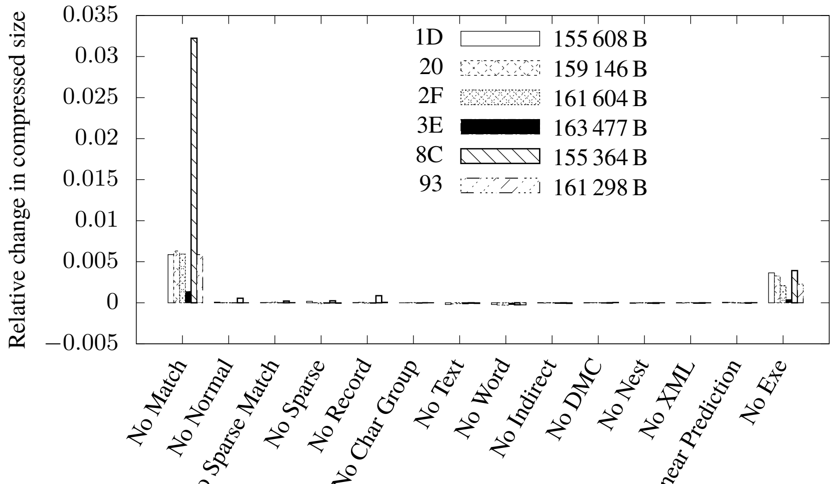
Figure 3.16.: Bitmap of first 4096 bit of selected devices of SRAMxmc dataset as 64 by 64 pixels with 1 as white, 0 as black. Devices 2A and 4A are well compressible by the sparse model, devices 1D and 20 are weakly compressible by the match model.

by the exe model. The exe model presumably tries to find common assembler instructions, which may coincide with the 32 bit alternating bias pattern that is a known issue of the dataset, cf. Sec. 1.9.4, since it tends to produce the same 32 bit words multiple times. Repeated 32 bit words may be even better captured by the match model, although the exact way it produces predictions has not yet been analyzed. For the low entropy devices such as 2A and 4A, see Fig. 3.17a, removal of the sparse model has the worst impact on compressed size. This is plausible, because these devices have only few response bits that divert from the alternating pattern, cf. Fig. 3.16, meaning the response consists mostly of 32 1s followed by 32 0s. Removal of any other model improves compressed size, which suggests that the sparse model is the only fit one. This, however, does not seem in line with the fact that change in compressed size is an order of magnitude smaller than removing the match model for one of the high entropy devices.

In general, the change in compressed size for any single model disabled is small for both the common text examples as well as the PUF candidate response data. The reason is that the combination of multiple models, even if unfit individually, can give quite good predictions when combined, which makes the context mixing concept so successful. Additionally, PAQ contains a so-called secondary symbol estimation (SSE) stage between the neural network that combines the predictions of the models and the arithmetic encoder that produces the final output. The influence of SSE increases the less fit the models are, so in particular when none of the available models fits well, it may strongly improve compressed size. By example of “Alice in Wonderland”, to turn SSE off while all models are enabled increases compressed size to 33 674 B, which is a relative increase by ≈ 0.007. With only the exe model enabled, though, compressed size is 93 794 B if SSE is enabled, but 148 576 B without SSE [93]. Here the lack of SSE causes a relative increase by ≈ 0.584.



(a) Devices with small absolute compressed size, i.e. low entropy.



(b) Devices with large absolute compressed size, i.e. high entropy.

Figure 3.17.: Change in compressed size with individual models disabled relative to compressed size with all models in PAQ enabled for true responses of selected devices from SRAMxmc dataset. Absolute compressed size with all models in PAQ enabled is printed next to the key. Lower is better.

So in conclusion, it seems worthwhile future work to customize a state-of-the-art statistical compressor with SSE for entropy estimation of PUF candidates and detection of particular statistical flaws. It should contain models of typical flaws found in PUF candidates, such as biased response bit positions or machine learning models that produce predictions from challenges. To identify which flaw is present, it should report not only the compressed size but also information about which model's predictions where most accurate overall. Until such is available, though, general purpose statistical compressors such as PAQ or CMIX may provide tighter bounds on entropy than CTW and should therefore be considered in future work.

3.9. Applicability of TRNG Test Suites

Apart from the PUF oriented metrics outlined in Chpt. 2, some authors chose to apply test suites for RNGs on the outputs of their PUF candidate, e.g. [35, 99]. To understand the implications of this, it is necessary to understand how such test suites work. The following discussion will use USA's national institute of standards and technology (NIST) SP800-22 [100] as an example, but other test suites such as AIS 31 from Germany's federal office for information security (BSI) [101] work similar. These test suites consist of a collection of *statistical hypothesis tests* that aim to reject the claim that their input indeed originates from a memoryless $\mathfrak{B}(1, p = 0.5)$ source.

Statistical hypothesis testing works by definition of a set of mathematical operations to be applied on the observations together with a decision rule when to reject a hypothesis or not. Here, the claim of the RNG to be a memoryless $\mathfrak{B}(1, p = 0.5)$ source is the null hypothesis H_0. The mathematical operations to be performed on the observations can be any that make their result follow a calculable distribution if the null hypothesis is true. For convenience, this is typically a well-known distribution such as normal, binomial, etc. To formulate the decision rule, a significance level, canonically denoted α, has to be decided upon before the operations are applied on actual observations. This is a highly intuitive task and the most popular guidance for this step is that virtually all statisticians chose either 1% or 5% for more than a hundred years. There are multiple objectives to way against in this step. On the one hand the false positive rate, which equals α, so the risk to reject the null hypothesis due to bad luck although it actually is true, which is called a type I error. On the other hand are two things: First, the usefulness of the test, since a test that never rejects the null hypothesis due to an extremely low α is useless. Second, the risk of a false negative, i.e. to not reject the null hypothesis although it is incorrect, referred to as type II error. The false negative rate is canonically denoted β and typically increases when α is decreased, but does not have a direct mathematical relationship to it, because it depends on how the result of the test is distributed under the true hypothesis, which is unknown. The last point also means that it in general is not possible to prove the null hypothesis, except for specific cases, because a test can reasonably only consider a single facet of the overall model, and it might be that the test result follows a very similar distribution under the true hypothesis although it is fundamentally different from the null hypothesis.

Once the significance level is decided upon, there are two ways to proceed that are both statistically sound and differ only in their practicality. If the test is to be performed many times and it only matters whether the null hypothesis should be rejected for an observation or not, it is advisable to come up with a table or decision rule on the result of the mathematical operations itself, e.g. "reject the PUF candidate if the computed number is greater than 5 or less than -3". This saves the effort to compute a p-value, which is the other way to proceed, but does not provide further information beyond the reject. A p-value instead represents the probability that, while the null hypothesis holds, this or a more extreme test result is observed. It can be directly compared to α, where the null hypothesis is to be rejected when the p-value is lower. In contrast to the previous approach, it also provides information on how clearly the null hypothesis is to be rejected. If the p-value is just barely below α, it may be reasonable to repeat the test with a larger dataset. Either way the statistical hypothesis test is completed by a reject or non-reject of the null hypothesis.

Since a statistical hypothesis test can only reject the null hypothesis, but rarely prove it or any other hypothesis, the test suites consist of a series of tests, where each test checks a different and ideally orthogonal facet of what is to be expected under the common null hypothesis. SP800-22 and most other suites start with a frequency test that has a relatively simple calculation and decision rule. It is based on the facet that if the output stems from a memoryless $\mathfrak{B}(1, p = 0.5)$ source, the overall number of 1s and 0s to be encountered in the output should be approximately equal. More formally, the number of 1s would be $\mathfrak{B}(N, p = 0.5)$ distributed, with N the length of the sequence. If there are only very few 1s or only very few 0s in the output, it can already be justified to reject the null hypothesis. However, it remains unknown which other hypothesis should be adopted instead: It might be that $p = 0.5$ in fact holds, but the source is not memoryless, or to the contrary that the source is memoryless, but has a different value of p. If otherwise the test has been passed, it might have been because the source produced $N/2$ 0s followed by $N/2$ 1s, which a human would intuitively dismiss as non random, but would perfectly pass the first test. Therefore, SP800-22 continues with a block frequency test, which does the same as the above after cutting the sequence into equally sized blocks. This test would catch above sequence of $N/2$ 0s followed by $N/2$ 1s, but then one may come up with e.g. a sequence of alternating bits such as 010101..., which would pass both tests so far with distinction. So another test is necessary to catch these sequences, which may again be passed by a more sophisticated yet predictable sequence, and so on and so forth. In the end, SP800-22 contains a total of 15 tests that were deemed appropriate and sufficient by NIST to detect any statistical defects in the output sequences that an attacker might use to predict earlier or later outputs of the RNGs.

While this concept of testing for signs of predictability seems well suited for PUFS, care has to be taken. RNGs produce one-dimensional data and the tests are supposed to carefully check for predictability within this one dimension, i.e. from the current RNG output to previous or later RNG output. They are not designed to take auxiliary information into account, such as the challenges that produced the output, or the helper data required for error correction. They are also not designed to check for predictability along multiple dimensions. While the tests accept multiple sequences, each of them is tested separately, so if e.g. a sequence is the response of a device, and the responses from multiple devices are fed into the test suite as multiple sequences, the test suite will check for each device whether knowledge of parts of its response can be used to predict other parts of its response, but it will not check for predictability across devices, e.g. that the third bit position tends to the same value on all devices. The latter would require to construct sequences across devices, e.g. so that a sequence contains the b^{th} response bit from all devices and there are as many sequences as there are response bit positions, and every sequence is as long as there are devices in the dataset. This approach, however, raises the question of how to order the devices within the sequences, because in contrast to mean or variance, where the order of samples is irrelevant, *all* but the frequency test in SP800-22 care most about the *order* of the 1s and 0s than about their amount. They must not be *sorted* by their response, of course, but could for example be ordered by serial number, or the order they were tested in. If the PUF candidate is unpredictable, the order would not matter, but it might be that predictability remains undetected because the order that reveals it was not tested. The situation becomes even more complicated when a multi-challenge PUF candidate produces multi-bit responses, because to obtain a one-dimensional sequence per device, the dimensions of challenges and response bit positions would need to be merged into one, for which there is no natural order, though. So the way multidimensional data is flattened is a critical question that might all alone decide whether a PUF candidate passes or fails, but for sure has an influence on the result of individual tests, as exemplified in Tbl. 3.4.

The first two lines in Tbl. 3.4 show the results for the ROmaiti dataset in two different orders. For both of them the pass rates of some tests are at an acceptable level, but overall too many tests fail to consider the dataset free of flaws. For the Ahori dataset in the following two lines, the order affects only the block-runs and serial test, while the non-overlapping template test passes for all devices in both

Sequence description	Sequence length	Monobit	Block-Frequency	Runs	Block-Runs	Matrix Rank	Non-Overlapping Templates	Serial	Cumulative Sums	No. Sequences
ROmaiti MEPWC, sequences from b^{th} response bit position on all devices, sorted by serial number	193	141	183	254	167	–	198	218	147	/256
ROmaiti MEPWC, sequences from entire response of a device	256	178	191	188	149	–	156	191	184	/193
Ahori, sequences by concatenation of l^{th} bit position of all identifiers	131072	0	0	–	9	0	45	12	0	/45
Ahori, sequences by concatenation of all identifiers	131072	0	0	–	37	0	45	3	0	/45
SRAMxmc 2015, first 1024 bit only, sequences across devices	144	654	815	1014	623	–	811	985	676	/1024
SRAMxmc 2015, first 1024 bit only, sequences within devices	1024	143	73	95	115	–	139	108	143	/144

Table 3.4.: Effect of different ways to construct one-dimensional RNG sequences from multi-dimensional PUF data on pass rates for eligible tests of NIST SP800-22 RNG test suite. The approximate entropy test would also be eligible, but is incorrectly implemented in version 1.2.0 of the `nistrng` Python module and therefore not reported.

orders, but the remaining tests fail for all devices and orders. The SRAMxmc dataset shows a similar behavior as the ROmaiti dataset, where the order affects all tests and the pass rates are acceptable for the runs test with sequences across devices, but otherwise only mid range.

To resolve this issue, two options could be followed: First, several different ways of flattening the data may be performed and the results of the test suites on each of them analyzed. This approach does not require changes to the tests, but increases computational effort. If the PUF responses are unpredictable, at most a small portion of tests that corresponds to the expected false positive rate may fail. If more tests fail, the PUF is to be considered flawed. Second, one may carefully analyze the individual tests and see if they can be extended to multiple dimensions, or at least tweaked so e.g. the order of devices becomes irrelevant to the test output. The latter approach is used in the following chapter, which introduces several hypothesis tests to evaluate the unpredictability of PUF candidates.

4. Improvements and Novel Metrics

One of the findings of the preceding chapter was that statistical hypothesis testing can be a solution to overcome subjective judgement whether metric results are close to their optimum or not. This chapter therefore exemplifies how a selection of common metrics can be extended into hypothesis tests in order to establish this approach as new standard for PUF evaluation. The set of tests is then expanded to detect spatial autocorrelation, a type of flaw that is not covered by any previously used metric in the PUF domain.

Apart from hypothesis tests, the response mass function is introduced, which for the first time allows to represent the probability distribution of the entire PUF response even for realistic response sizes of hundreds or thousands of bit. This allows to extend the expected conditional min-entropy, which remains about a PUF response if the attacker knows the probability distribution and the actual helper data, to multivariate statistical models.

4.1. Hypothesis Tests From Common Existing Metrics

The statistical models introduced in Sec. 3.5 enable to construct statistical hypothesis tests from selected common and widely used existing metrics. The power of these tests is yet to be researched, however the extension of well-known metrics into hypothesis tests is believed to help establish a new basic approach to the evaluation of PUF candidates, where metrics no longer provide numeric results without context, but well understood evidence in favor or against unpredictability. It furthermore makes it easier to understand how these tests work and what defects they are able to flag. It may even be possible to reassess the results of previous papers, when their numeric values can now be interpreted in the light of expected distributions and decision criteria.

4.1.1. Distribution of Uniformity with Univariate and Multivariate Models

The HW of a device's response, defined by Maiti et al. [70] for single-challenge PUFs as Uniformity, lends itself to be extended into a hypothesis test, because the statistical models introduced in Sec. 3.5 are constructed in a way to represent certain response bit positions, challenges, or a combination thereof as RVs, while devices represent samples of these RVs. The HW of the response may thus serve as a test statistic to verify or reject the null hypothesis H_0. The null hypothesis here is that the response of a device consists of a set of independent samples of these RVs with no further dependencies except those included in the selected statistical model.

Once the distribution of the test statistic under H_0 is found, two approaches can be followed. The first is to test each device's HW individually against the expected distribution and calculate a p-value per device. As long as the number of devices that fail at a given significance level is about the number of expected false rejects, there is not sufficient evidence to claim the PUF candidate circuit is flawed. The second approach is to use a goodness of fit test to check if the observed HWs do follow the expected distribution. Since the expected distribution of HW is discrete, common goodness of fit tests such as the Anderson-Darling or Kolmogorov-Smirnov test are inapplicable. However, if the number of test devices is sufficiently large to have at least five devices in each bin and an adequate number of bins, a chi-squared test can be performed.

Expected Distribution Under Null Hypothesis

For the univariate Bernoulli model, the distribution of the test statistic under H_0 is rather simple. Since the probability for a 1 response is identical for all response bit positions, it is the sum of B Bernoulli trials with identical success probability and therefore $\mathfrak{B}(B, p)$ distributed.

For the multivariate Bernoulli model, the success probability may differ between RVs and therefore does not result in a binomial distribution. However, the PMF respectively PDF of the sum of multiple independent RVs equals the convolution of the individual PMFs respectively PDFs [102]. For that reason the expected distribution of the HW under H_0 can still be obtained, because the HW of the entire response is the sum of the HWs of individual response bit positions, which are independent under H_0. This may be exemplified by the following two simple cases: First, let a PUF candidate design have $B = 2$ and success probability vector

$\underline{p} = \begin{pmatrix} p_1 & p_2 \end{pmatrix}$ with $p_1 \neq p_2$, then the PMF of the HW T is

$$f_T(t) = \begin{cases} (1-p_1)(1-p_2) & \text{if } t = 0 \\ p_1(1-p_2) + (1-p_1)p_2 & \text{if } t = 1 \\ p_1 p_2 & \text{if } t = 2 \end{cases}, \tag{4.1}$$

which is equal to a convolution of the PMFs of the individual bit position's HW:

$$f_T(t) = \begin{pmatrix} (1-p_1) & p_1 \end{pmatrix} \circledast \begin{pmatrix} (1-p_2) & p_2 \end{pmatrix} \tag{4.2}$$

Second, if the multivariate Bernoulli model has less than B RVs, because there are multiple response bit positions assumed to have the same probability to turn out as 1, the HW of these response bit positions is binomial rather than Bernoulli distributed, but still contributes an independent summand to the entire HW. Let a PUF candidate design have $B = 4$, where the first three response bit positions have probability p_1 to turn out as 1 and the last response bit position turns out as 1 with p_2. The PMF of the HW T is

$$f_T(t) = \begin{cases} (1-p_1)^3(1-p_2) & \text{if } t = 0 \\ \binom{3}{1} p_1 (1-p_1)^2 (1-p_2) + (1-p_1)^3 p_2 & \text{if } t = 1 \\ \binom{3}{2} p_1^2 (1-p_1)(1-p_2) + \binom{3}{1} p_1 (1-p_1)^2 p_2 & \text{if } t = 2 \\ p_1^3 (1-p_2) + \binom{3}{2} p_1^2 (1-p_1) p_2 & \text{if } t = 3 \\ p_1^3 p_2 & \text{if } t = 4 \end{cases}, \tag{4.3}$$

which may again be written as a convolution of PMFs:

$$f_T(t) = f_{\mathfrak{B}(3,p_1)} \circledast f_{\mathfrak{B}(1,p_2)} \tag{4.4}$$

The resulting PMF has a narrower peak, i.e. smaller variance, than a $\mathfrak{B}(B, p)$ distribution, where p is the mean of the individual success probabilities p_i, which is known as the so-called binomial sum variance inequality. This means the responses of such a PUF candidate design tend to have similar HW. However, this is a minor issue compared to the increased collision probability in comparison to a PUF candidate design with homogeneous success probability p.

For the multivariate categorical distribution, the convolution approach applies equally, though the PMFs of individual RVs are neither Bernoulli nor binomial distributed, but categorical. Even a mixed model is possible for this test, where some parts of the response stem from the same Bernoulli distribution, while other parts are Bernoulli with a different success probability or follow a particular categorical distribution. For example, let a PUF candidate design have $B = 4$, where the first and second response bit position is Bernoulli with p_1, and the remaining two response bit positions are determined by $p_{00}, p_{01}, p_{10}, p_{11}$. The PMF of the HW T is

$$f_T(t) = \begin{cases} (1-p_1)^2 p_{00} & \text{if } t = 0 \\ \binom{2}{1} p_1(1-p_1)p_{00} + (1-p_1)^2(p_{01}+p_{10}) & \text{if } t = 1 \\ p_1^2 p_{00} + \binom{2}{1} p_1(1-p_1)(p_{01}+p_{10}) + (1-p_1)^2 p_{11} & \text{if } t = 2 \\ p_1^2(p_{01}+p_{10}) + \binom{2}{1} p_1(1-p_1)p_{11} & \text{if } t = 3 \\ p_1^2 p_{11} & \text{if } t = 4 \end{cases}, \tag{4.5}$$

which is again equal to a convolution of PMFs

$$f_T(t) = f_{\mathfrak{B}(2,p_1)} \circledast f_{\mathfrak{C}(0\mapsto p_{00}, 1\mapsto(p_{01}+p_{10}), 2\mapsto p_{11})}. \tag{4.6}$$

Test Evaluation

Sensitivity to Correlated Response Bit Positions A defect that may be detected by this test are correlated response bit positions. They will either increase variance among devices' HW if correlated positions tend to produce the same response, or reduce variance if correlated positions tend to produce the opposite response. To better understand this effect, consider a trivial two bit example without bias. If both response bits are independent, the probability for 00, 01, 10, and 11 are equally 0.25. Under positive correlation, the probability for 00 and 11 increases and the probability for 01 and 10 decreases accordingly. For the HW, this results in more extreme values, in this case 0 and 2, which become more probable than centered values, in this case 1. Under negative correlation, the probabilities change inversely, meaning that responses 01 and 10 become more probable while 00 and 11 become less probable. The effect on HW turns around the same way, now favoring centered values over extreme values. Note, however, that the effect of positive and negative correlation can cancel each other out as in other tests.

To evaluate the performance of the test in detection of such flaws, three artificial datasets with $D = 10\,000$ and $B = 1024$ have been crafted. In the first dataset, all responses are randomly and independently sampled from a fair Bernoulli distribution. For the second dataset, the responses of the first 256 response bit positions are copied into the last 256 response bit positions, so these positions are perfectly positively correlated. The third dataset has the last 256 response bit positions inverted, so that they are perfectly negatively correlated to the first 256 response bit positions.

All datasets are tested under the univariate Bernoulli model with success probability ≈ 0.5002, estimated from the overall ratio of 1 responses. The resulting PMFs and histograms of observed HWs are displayed in Fig. 4.1. As expected, the histogram of observed HWs becomes wider than a binomial distribution for positively correlated response bit positions and sharper for negative correlation. To obtain p-values for individual devices, the CDF and the survival function (SF), which equals the difference between unity and the CDF, are evaluated at the observed HW. The lower of both probabilities is then doubled to account for the fact that the test is two-sided. The approach to select the lower of both probabilities finds the p-value, i.e. the probability to observe this or a more extreme observation, independent of whether the observed HW is in the left or right tail and independent of the success probability. For a significance level of $\alpha = 0.05$, 482 devices with independently sampled response bits fail, which is approximately the expected number of $\alpha D = 500$ false rejects. Both correlated cases are detected as the number of failing devices increases to 1082 with positive correlation and reduces to 42 in case of negative correlation.

To verify the results, a chi-squared test is performed in addition, where bin edges are chosen so that for each bin at least $^{D}/_{50}$ observations are expected and each bin contains at least $^{D}/_{50}$ actual observations. For the independently sampled case, this results in 38 bins and a chi-squared result of 27.6, which relates to a p-value of 0.84, so there is no reason to reject the assumption that the observed HWs indeed follow the distribution under H_0. For the positively correlated case, the number of bins is 30 and the chi-squared result is 1045, which relates to a p-value of $\approx 10^{-200}$ and therefore the observed HWs clearly do not follow the binomial distribution expected for uncorrelated data. The same holds for the negatively correlated case, which produces a chi-squared result of 1538 from 33 bins.

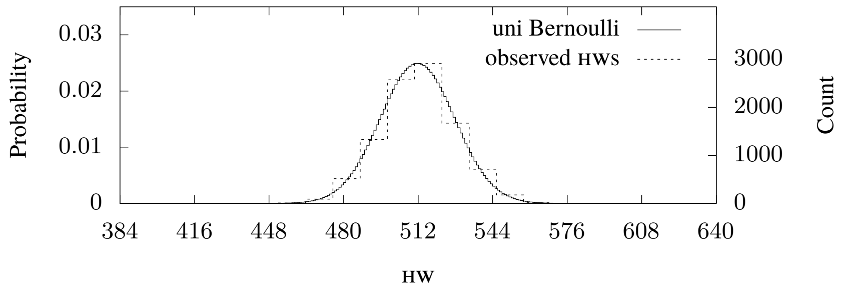
To validate that correlations such as those used in this example can be represented in the multivariate categorical model, the correlated datasets are additionally tested against the expected distribution in this case, i.e.

$$f_T(t) = f_{\mathfrak{B}(512,p')} \ast \left(\mathop{\ast}_{i=1}^{256} \begin{pmatrix} 1-p'' & 0 & p'' \end{pmatrix} \right) \tag{4.7}$$

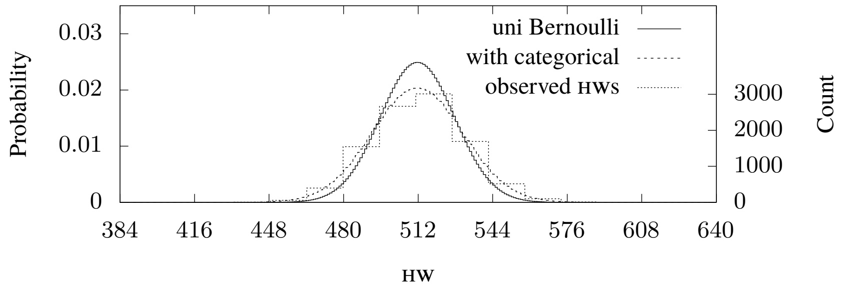
for the case of perfect positive correlation and

$$f_T(t) = f_{\mathfrak{B}(512,p')} \ast \left(\mathop{\ast}_{i=1}^{256} \begin{pmatrix} 0 & 1 & 0 \end{pmatrix} \right) \tag{4.8}$$

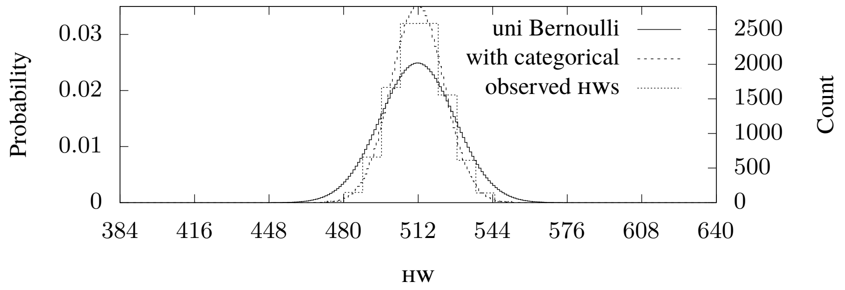
for the case of perfect negative correlation. $p' \approx 0.5005$ is estimated from the ratio of 1 responses among uncorrelated response bit positions 257 to 768, while $p'' \approx 0.5000$ is estimated from the ratio of 1 responses among the first 256 response bit positions. As visible in Fig. 4.1, the so obtained distributions match



(a) IID unbiased response bits.



(b) Positive correlation, i.e. responses on the last 256 bit positions are identical to those on the first 256 positions.



(c) Negative correlation, i.e. responses on the last 256 bit positions are the inverse of those on the first 256 positions.

Figure 4.1.: Uniformity for a crafted dataset with artificial positive, negative, and without correlation and PMFs of expected distribution with and without consideration of correlations. PMFs use the y-axis on the left hand side, bins of observed HW use the y-axis on the right hand side.

the histograms of observed HWs more closely and the number of devices that fail the hypothesis test at the $\alpha = 0.05$ significance level are now 466 for the second and 440 for the third dataset, which both are much closer to the expected 500 false rejects than without modeling the correlation. Chi-squared tests against these distributions with 33 bins in both cases pass with chi-squared results of 23.9 respectively 36.8, which correspond to p-values of 0.78 and 0.18. This verifies that the effect of correlations on the observed HWs can indeed be modeled as described above.

Given that correlated response bit positions can be detected by a test against the univariate Bernoulli model, one may question how this approach compares to other methods such as Pearson's correlation coefficient. A benefit of this approach is that it does not require knowledge about *which* response bit positions are correlated or otherwise dependent. To inspect the correlation coefficient among all pairs of response bit positions would require $\binom{1024}{2} = 523\,776$ calculations between 10 000 element sequences. Among the downsides is that the particular reason why the distribution of HWs does not follow the expected one remains unknown. It might be due to correlations, but could also be due to varying success probability, which means a multivariate Bernoulli model rather than a univariate model would be required. Furthermore, the effect of positive and negative correlation among different response bit positions may cancel each other out in the HWs, so their observed distribution does not sufficiently deviate from a binomial distribution to be detected. The test can therefore only serve as part of a larger test suite.

ROmaiti As an example with a real-world dataset, the ROmaiti dataset with response bits extracted by MEPWC is evaluated under a univariate and multivariate Bernoulli model. For the univariate Bernoulli model, the number of trials of the binomial distribution is $B = 256$ and the success probability is estimated as the overall ratio of 1 responses

$$p = \frac{1}{BD} \sum_{d=1}^{D} \sum_{b=1}^{B} \bar{x}_{b,d} \tag{4.9}$$

to ≈ 0.478. The resulting PMF is plotted in Fig. 4.2 together with the PMF under the multivariate Bernoulli model. For the latter, one RV per response bit position is used, so the PMF results from convolution of B Bernoulli PMFs with success probability

$$m_b = \frac{1}{D} \sum_{d=1}^{D} \bar{x}_{b,d}. \tag{4.10}$$

The effect of the binomial sum variance inequality can be seen as the PMF under the multivariate Bernoulli model is narrower and has a higher peak than

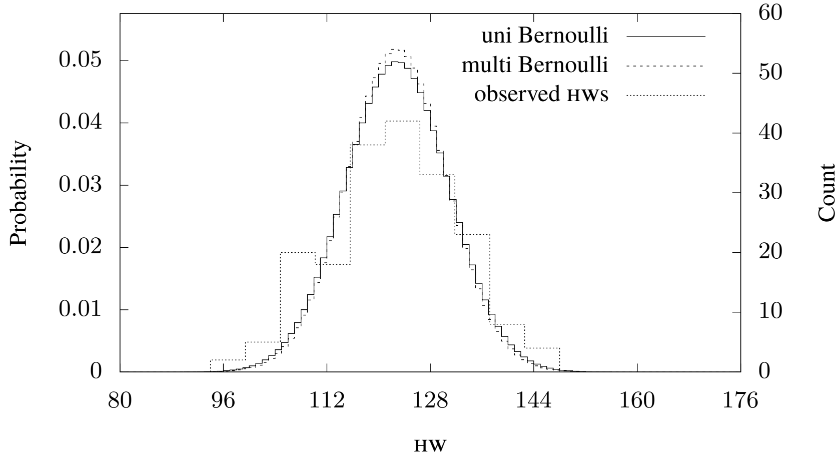


Figure 4.2.: Uniformity for ROmaiti dataset with response bits from MEPWC and expected PMFs under univariate and multivariate Bernoulli model.

the PMF under the univariate Bernoulli model. However, the difference between both distributions is small. To obtain p-values, the minimum of CDF and SF at the observed HW is doubled, as done in the previous example. Under both models, 21 out of 193 devices fail the hypothesis test at the $\alpha = 0.05$ significance level, which is about twice as many failures as false rejects are to be expected. Chi-squared tests with 21 bins against the expected distributions under the univariate and multivariate model produce 33.7 and 46.0, which relate to p-values of 0.0198 and $4.90 \cdot 10^{-4}$. This suggests that a multivariate Bernoulli model does not fit better to the ROmaiti dataset than a univariate Bernoulli model, and neither model appropriately resembles the dependencies that seem to affect the HWs most.

4.1.2. Distribution of Bit-Alias for Univariate Model

Under the univariate Bernoulli Model, the HW of a response bit position among all devices, known as Bit-Alias by Maiti et al. [70], is expected to be binomial distributed, similar to Uniformity described in the previous test. The null hypothesis for the corresponding test is again that all response bits are drawn from the same univariate Bernoulli source and no other dependencies exist in the data. The Bit-Alias should then follow a $\mathfrak{B}(D, p)$ distribution, where D is the number of devices and p may be estimated from the observed responses. Similar to above, one may either test individual Bit-Alias against the expected distribution and verify that only about the number of false rejects fail for a given significance level, or the

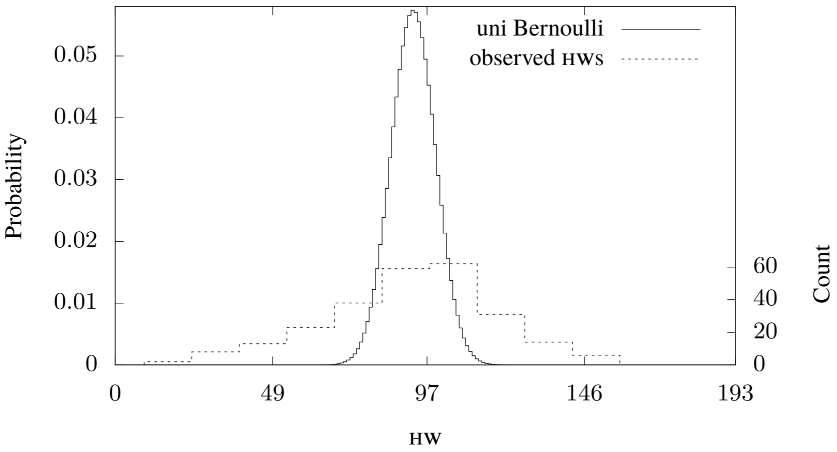


Figure 4.3.: Bit-Alias for ROmaiti dataset with response bits from MEPWC and expected PMFS under univariate Bernoulli model with $p = 0.478$.

entire distribution of Bit-Alias is tested against the expected binomial distribution by a goodness-of-fit test such as the chi-squared test. If the Bit-Alias do not follow a binomial distribution, to properly model them either requires a multivariate Bernoulli model because the success probability is not identical, or correlations or other dependencies mandate a categorical model.

To exemplify this test, the ROmaiti dataset is used again, because the results from the hypothesis test on Uniformity suggest a univariate Bernoulli model fits slightly better than a multivariate Bernoulli model. A histogram of the observed Bit-Alias values and the PMF of the expected binomial distribution if the univariate Bernoulli model would apply are depicted in Fig. 4.3. As clearly visible, the Bit-Alias do not follow the expected distribution. Instead of an expected number of around 10 response bit positions that are incorrectly rejected for $\alpha = 0.05$, 151 response bit positions fail at this significance level. A chi-squared test as performed for the previous test on Uniformity produces a result of 1717 from 15 bins, where the corresponding p-value is reported as zero due to numerical limitations.

This provides a welcomed example of how multiple tests can complement one another to provide a more complete picture. Based on the results of both tests, at least a multivariate categorical model is required to appropriately model the ROmaiti dataset. A univariate Bernoulli model is unfit since this test shows that the success probability among response bit positions is clearly not identical. A multivariate Bernoulli model is unfit since it would reduce the variance of

Uniformity, while the corresponding test reveals increased variance, which hints at positive correlation between response bit positions.

4.1.3. Permissible Bias

Typical hypothesis tests for bias, such as the monobit test in the TRNG test suite by NIST [100], have a null hypothesis of unbiasedness that is rejected if the tested RNG sequence contains too few or too many 1s. This way, however, one may find reason to reject the hypothesis of no bias, but cannot find reason to reject accusations of bias. The hypothesis test described in the following therefore has a null hypothesis of bias stronger than a given permissible threshold, which is to be rejected if the tested sequence appears free of bias. This construction is particularly helpful if e.g. the post-processing algorithm for the PUF response is able to correct up to a certain amount of bias among all response bits in a univariate Bernoulli model or the security evaluation requires the assumption that no response bit position suffers from more than a given amount of bias towards either outcome in a multivariate Bernoulli model. In the multivariate case, this test may be regarded as an extension of the Bit-Alias metric by Maiti et al. [70] into a statistical hypothesis test. A preliminary version of this section has been published in [**91**].

Based on the Clopper-Pearson method for a CI of the binomial proportion introduced in Sec. 3.7, a hypothesis test for unbiasedness can be constructed. Similar to the underlying tests of the CI, it consists of two hypothesis tests that both are supposed to be rejected to prove unbiasedness. Their null-hypotheses are $\mathrm{H}_0 : p \geq p_{\rhd}$ respectively $\mathrm{H}_0 : p \leq p_{\lhd}$, and their combined alternative hypothesis is $\mathrm{H}_\mathrm{A} : p_{\lhd} < p < p_{\rhd}$. With the assumption that the sequence to be tested originates from the same Bernoulli source, the number of 1 responses is $t \leftarrow T \sim \mathfrak{B}(N, p)$, where N is the length of the sequence and may equal D for the multivariate case or BD for the univariate case. The null-hypotheses may thus be rejected if the observed number of 1 responses is too low respectively too high under the respective null hypothesis. Note that due to the monotonicity of CDFs it is sufficient to consider the hypotheses $p = p_{\rhd}$ respectively $p = p_{\lhd}$, because if one or both of them is rejected, so will be the hypothesis $p = p' \quad \forall p' > p_{\rhd}$ respectively $p = p' \quad \forall p' < p_{\lhd}$. The p-values for both tests can be obtained from

$$p_{0,\rhd} = \mathrm{P}[T_{\rhd} \leq t] = \sum_{i=0}^{t} \binom{N}{i} p_{\rhd}^{i} (1 - p_{\rhd})^{N-i} \tag{4.11}$$

respectively

$$p_{0,\triangleleft} = \mathrm{P}[T_{\triangleleft} \geq t] = \sum_{i=t}^{N} \binom{N}{i} p_{\triangleleft}^{i} (1 - p_{\triangleleft})^{N-i}, \tag{4.12}$$

where $T_{\triangleright} \sim \mathfrak{B}(N, p_{\triangleright})$ and $T_{\triangleleft} \sim \mathfrak{B}(N, p_{\triangleleft})$. Since both tests need to reject their null hypothesis simultaneously to result in the common alternative hypothesis, both p-values are required to be less than $\alpha/2$ to achieve a common false acceptance rate, i.e. incorrectly accepting the alternative hypothesis of at most a permissible amount of bias, to be α. For $\alpha = 0.01$ and $p_{\triangleleft} = 0.45$, $p_{\triangleright} = 0.55$, it requires a minimum of $N = 680$ observations to achieve this with $t = 340$.

The probability to observe $t = 340$ in this case is even for $p = 0.5$ only 0.03, though. Practical application of the test therefore requires additional observations to achieve an acceptable false rejection rate. If $t_{\triangleright}$ and $t_{\triangleleft}$ are the maximum and minimum number of observed 1 responses that cause rejection of both null-hypotheses, the probability of a sequence with success probability p to be accepted is

$$p_{\mathrm{A}} = \mathrm{P}[t_{\triangleleft} \leq t \leq t_{\triangleright}] = \sum_{i=t_{\triangleleft}}^{t_{\triangleright}} \binom{N}{i} p^{i} (1 - p)^{N-i}. \tag{4.13}$$

Although this probability approaches unity for $N \to \infty$ if $p_{\triangleleft} < p < p_{\triangleright}$, the required number of observations for a practical application would be inferred from a chosen false rejection rate β that should be maintained within a shorter range of success probability $(p_{|\triangleleft}, p_{\triangleright|})$, where $p_{\triangleleft} < p_{|\triangleleft} < p < p_{\triangleright|} < p_{\triangleright}$ and $\forall p \in (p_{|\triangleleft}, p_{\triangleright|}) : 1 - p_{\mathrm{A}} \leq \beta$. This means within the range of acceptable bias, another range of expected bias is defined, and the probability that a device within the range of expected bias is incorrectly rejected should be β.[1] In the above example with $\alpha = 0.01$, $p_{\triangleright} = 0.45$, $p_{\triangleleft} = 0.55$, it would require $N = 6674$ to maintain $\beta = 0.01$ within $p_{|\triangleleft} = 0.48$, $p_{\triangleright|} = 0.52$.

Since the test is based on the Clopper-Pearson CI, the same decision can be reached by comparison of the CI limits to $p_{\triangleleft}, p_{\triangleright}$, but this would require to calculate the CI for every observed t individually. With the test, limits on t can be inferred, which saves the effort to calculate CIs in particular for the multivariate case, where many Bit-Alias have to be tested.

[1]This is analogous to electronic filter designs, where a pass band with a maximum attentuation and a stop band with a minimum attenuation is defined. The shorter the transition region and the higher the difference in attenuations, the higher the required order of the filter. For the hypothesis test, this means the shorter the distance between the limits of acceptable and expected bias, the more samples are required to maintain α and β in both.

4.2. Tests For Spatial Artifacts

An issue that is well known in the field of analog IC design, but which has been only rarely mentioned in previous work from the field of PUFs, is that of spatial artifacts in the manufacturing variations. Examples of such artifacts include edge effects, which means PUF cells at the edge of an array tend to a certain response due to a change in surrounding structure and can be alleviated through dummy cells around the array that do not contribute to the response [6]. They also include gradient effects, e.g. of oxide thickness, that extend over the whole wafer and cause e.g. ROs to tend to a higher frequency in some areas and lower frequency in other areas, where the location of fast and slow areas may differ on each die [**71**]. Spatial artifacts may also occur during readout, e.g. when oscillation based PUF types lock-in to the same frequency if they are physically proximate [23]. Another conceivable case is coupling via the supply grid for inverter based PUF types such as SRAM, where the current that a PUF cell draws until it reaches its final state influences the supply voltage, and thus possibly response, of surrounding cells because of the non-zero resistance of the supply lines.

In the few works that do mention spatial artifacts, their existence and consequence is assessed different. Su et al. mentioned the issue and stated that therefore "systematic offsets, process gradients, and shadowing effects must be minimized, which necessitates the use of analog layout techniques" [6] and made them compare a symmetric and a common-centroid layout of their SRAM-like ASIC design. Holcomb et al. [66] instead deemed spatial artifacts not relevant, because according to them, threshold voltage is the only relevant parameter for the response of their SRAM PUFs and previous work from the field of IC design is summarized to show that variations "in threshold voltages are due to random fluctuations in the concentration of dopant atoms, and are not spatially correlated" [66].

The term *spatial correlation* is used without further explanation in [66], however the context suggests it refers to *spatial autocorrelation* in the statistical sense. The latter describes a subset of spatial artifacts where samples from *neighbouring* locations affect each other. If they tend to be more similar than expected for independent samples, it is called *positive* spatial autocorrelation, and correspondingly if they tend to be less similar it is referred to as *negative* spatial autocorrelation. While other forms of spatial artifacts, such as edge effects, may be easy to spot in a layout preserving plot, because they form a geometric shape the human eye is trained to see, spatial correlations may remain imperceptible except for strong cases. To provide a qualified answer also in these cases, formal hypothesis tests on spatial autocorrelation exist in the field of statistics. A comprehensive introduction of these kind of tests to the field of PUFs has been provided in [**60**] and is contained in Sec. 4.2.2.

Apart from the question whether a given plot contains spatial artifacts or not, it is also important to distinguish which plot to look at, i.e. in which statistical figure the spatial artifact shows up. Three groups can be formed here:

The first group comprises artifacts that affect the same PUF cell on every device, such as edge effects or gradients of fixed direction. They are easily flagged by tests that follow – intentionally or not – the multivariate Bernoulli distribution approach, i.e. that investigate the response of each PUF cell among all devices, because they affect the estimated success probability. This is commonly referred to as bias, in particular if it causes the response of a PUF cell to differ from the optimal uniform distribution. One of the most intuitive tests for this group is a surface plot or heatmap of each bit position's FHW along devices, cf. (2.33), according to the on-die layout of the PUF cells. Examples of this are found in e.g. [6, 56, 67].

The second group is no longer visible as a bias, but rather a form of correlation. An example are gradient effects where direction and slope of the gradient changes from device to device. These changes may compensate each other so that there is no individual location on the die where the corresponding response bits from all devices would show a significant deviation from a uniform distribution. Instead, it requires to investigate how the response bits from multiple locations relate to each other among devices. Still, it might be difficult to spot gradient effects in a common correlation matrix plot, so either problem specific preprocessing is necessary or advanced methods such as a Fisher-Yates test for independence [**37**] or a principal component analysis (PCA) [**37**, **71**] is necessary. The application and advantages of the latter are described further in Sec. 4.2.1.

In the third group, the spatial artifacts arise in each device's response in a particular way, so that artifacts may neither show up as a bias nor as a correlation. An example could be the coupling through leakage current or supply grid. To inspect each device's response in a layout preserving plot is a tedious and error prone task, especially since flaws in this group are more likely to cause spatial correlation rather than a clear geometric pattern. Artifacts of this group are thus in practice only identifiable by looking at the distribution of SPACA results, since they test each device individually and provide a numeric value that represents how probable this device's response is to be affected by spatial autocorrelation. See Sec. 4.2.2 for details and examples.

Note that the consequence of spatial artifacts on the unpredictability of a PUF candidate circuit depends also on the type of bit extraction used. For example, in the case of RO PUFs with gradient effects in frequency response data, an MEPWC of adjacent ROs works as a spatial high-pass filter, which reduces the ability to detect gradient effects in the binary responses, but also the impact of a gradient in RO frequencies on the security of the binary response. In this situation it might be worthwhile to investigate not only binary response data, but also the underlying continuous response values. This section therefore makes an exception to the general focus of this work on metrics for binary responses and presents PCA and SPACA for both cases, continuous and binary. Most of the formulae that describe the methods apply identically to both cases and ξ may readily be replaced with x. Where distinction is necessary, this is explicitly noted.

Since spatial artifacts typically appear where the response originates from an array of PUF cells and it makes the concept easier to grasp, this section will operate mostly within the dimension of response bit positions b, and – if necessary – the dimension of devices d. Although b has been defined in Sec. 1.8 to iterate through positions in the response bit string, it is used in this section to iterate through PUF cell locations under the assumption that a bijective relationship between both exists. This holds if each PUF cell contributes a single response bit. Where this is not the case, it may be re-established by considering the response bit string a string of multi-bit symbols, where b iterates through symbols rather than bits.

Analysis for spatial artifacts within the challenge space is currently not supported, so index c is omitted for brevity in this section. An extension into this direction requires to solve the issue of sparse observations, since for any practical PUF it must be infeasible to fully evaluate the challenge space. So either a tiny fraction of the challenge space is fully analyzed, which would ignore artifacts anywhere else, or the entire challenge space is covered in an equidistant way, which contradicts with the assumption that dependence is due to proximity, because the distance of the challenges, in a yet to determine distance metric, would be large considering the typical size of a challenge space and a feasible number of challenges to analyze. Research into this direction is left to future work.

As spatial artifacts affect unpredictability rather than reliability, the methods described in this section are typically applied on the true response, which allows to furthermore omit indices a, e.

4.2.1. Principal Component Analysis

PCA is a linear transformation often used in explorative analysis of multivariate problems. It has been introduced to the field of PUFs in [**71**]. Given a sample dataset, where each apparent RV constitutes a column, or its correlation matrix, it performs a basis transformation onto a new orthonormal basis, so the transformed RVs are uncorrelated. It uses the eigenvectors of the correlation matrix as new basis and sorts them by the amount of variance they explain in the original dataset. So the first principal component is the eigenvector that would resemble the most variance if the whole dataset where to be explained by a single RV. In other words, it minimizes the remaining variance, so it is a least-squares fit of a straight line into the dataset. The second principal component then is the eigenvector that would explain most of the remaining variance, and so on.

While the transformation can be performed from the sample dataset's correlation matrix alone, it is typically obtained from a singular value decomposition (SVD) of the sample dataset itself after standardization, i.e. from its z-scores. Using a dataset $\underline{\underline{\bar{x}}}$ or $\underline{\underline{\bar{\xi}}}$ with the true responses – either binary or continuously valued – of a single-challenge PUF as example, the z-scores standardized along devices $\overset{\langle d\rangle}{\underline{\underline{\bar{\xi}}}}$ are calculated as

$$\overset{\langle d\rangle}{\bar{\xi}}_{b,d} = \frac{\bar{\xi}_{b,d} - m_b}{s_b}, \text{ where} \tag{4.14}$$

$$m_b = \frac{1}{D} \sum_{d=1}^{D} \bar{\xi}_{b,d} \text{ and} \tag{4.15}$$

$$s_b = \sqrt{\frac{1}{D-1} \sum_{d=1}^{D} (\bar{\xi}_{b,d} - m_b)^2} \tag{4.16}$$

are the sample mean and sample standard deviation, respectively. To apply the SVD for PCA, a column must represent a RV and a row an observation, so $\overset{\langle d\rangle}{\underline{\underline{\bar{\xi}}}}$ has to be transposed before the SVD provides the relationship

$$\overset{\langle d\rangle}{\underline{\underline{\bar{\xi}}}}^{\top} = \underline{\underline{u}}\,\underline{\underline{v}}\,\underline{\underline{w}}^{\top}. \tag{4.17}$$

The meanings of these matrices are as follows:

- Columns of $\underline{\underline{w}}$ are the orthonormal principal components, also called loadings, showing conjointly behaving variables.
- Rows of $\underline{\underline{u}}\,\underline{\underline{v}}$ are the estimations of the principal components for each observation, i.e. device.
- Diagonal matrix $\underline{\underline{v}}$ contains the square roots of the eigenvalues of the covariance matrix of $\overset{\langle d\rangle}{\underline{\underline{\bar{\xi}}}}^{\top}$, which are proportional to the amount of variance in the dataset described by this principal component.

Note that due to the use of z-scores, a PCA is insensitive to spatial artifacts of the first group, i.e. those that affect the same PUF cells in the same way on every device, such as edge effects or gradients of fixed slope and direction. Instead, it detects flaws of the second group, which is exemplified in the following by application to the ROmaiti dataset, cf. Sec. 1.9.2, and a crafted dataset of the same size with standard normal distributed values. For both datasets, continuous response data as well as binary response data from multiple bit extraction methods is evaluated.

RO Frequency Data by Maiti et al.

Fig. 4.4a shows the correlation matrix for the true response frequencies of the ROmaiti dataset, i.e.

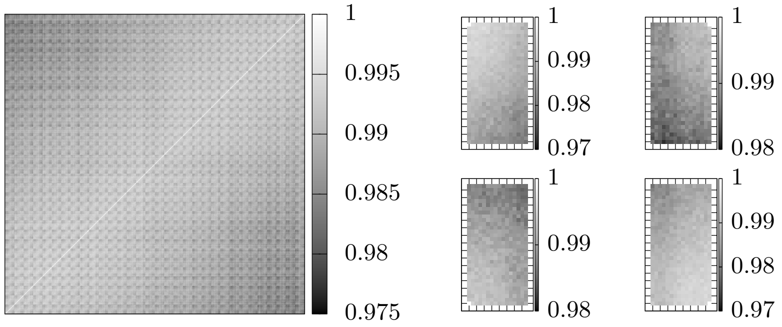
$$\bar{\xi}_{b,d} = \frac{1}{A} \sum_{a=1}^{A} \xi_{a,b,d}. \tag{4.18}$$

The correlation matrix is close to one for all elements, because all ROs on a die are equally affected by die-to-die variation, so there are devices where all ROs on all positions tend to be faster than the mean of their position among all devices and there are devices where all ROs tend to be slower. While this may be interpreted in a way that RO PUFs are useless, since all ROs behave identically, it in fact does not impair the performance, because any practical application will only compare ROs among the same device. If selected columns are plotted in a way that preserves the 16 by 32 on-die layout, a weak correlation that depends on the physical distance between the compared ROs becomes visible, but may be disregarded as measurement noise since its difference from one is very low in all cases. The picture becomes slightly better if the mean frequency of all ROs on a device is subtracted,

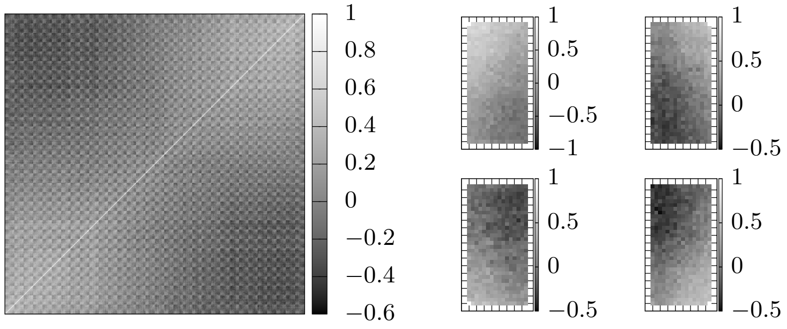
$$\bar{\xi}'_{b,d} = \bar{\xi}_{b,d} - \frac{1}{B} \sum_{b'=1}^{B} \bar{\xi}_{b',d} \qquad \begin{array}{c} b, b' \in \{1, \ldots, B = 512\} \\ d \in \{1, \ldots, D = 193\} \end{array}, \tag{4.19}$$

which has its correlation matrix, again together with selected elements in layout preserving form, shown in Fig. 4.4b. Although the spatial correlation in the dataset is already quite well identifiable now, subtraction of the mean frequency of all ROs on a device is a problem specific preprocessing that most likely would not be performed if there is not yet a suspicion for spatial correlation.

With a PCA on $\bar{\xi}_{b,d}$, a more obvious and more detailed result is obtained with a standardized method instead of problem specific preprocessing. Since the input data contains 193 observations for every RV, the output has 193 principal components. For each of the first eight principal components, Fig. 4.5 shows the so-called loading in a layout preserving heatmap, a histogram of the estimations, and the amount of variance described by it. The first principal component automatically captures the die-to-die variation, since it affects all ROs on a device in the same way (top left heatmap in Fig. 4.5). The second and third principal components capture the linear portion of the gradient effects, and the third to eighth components the nonlinear portion, since they still contain identifiable spatial patterns. Note that the clear north-south and east-west gradients in the second and third components are not by accident. Plotted in layout preserving way, it may seem that the PCA simply chose the y and x axis as components, but



(a) RO frequencies



(b) Deviations of RO frequency from mean frequency on a device

Figure 4.4.: Correlation matrix and selected elements in layout preserving form for RO frequencies (mean among accesses) in Maiti et al. dataset

the on-die layout is completely unknown to the PCA and only added for the plot. So whatever information made the PCA choose these principal components must be in the RO frequency data itself. Finally, the local manufacturing variations that are of interest for the unpredictability of the PUF are covered by approximately the ninth principal component onward. Since these variations are individual to a device, they do not follow a common pattern among all devices and are thus spread among multiple components due to the way PCA works.

Independent Standard Normal Distributed Data

To assess the behavior of a PCA for data without spatial artifacts and increase confidence in the previous results not being observed at random, PCA is here applied on a dataset entirely drawn independently from a standard normal distribution.

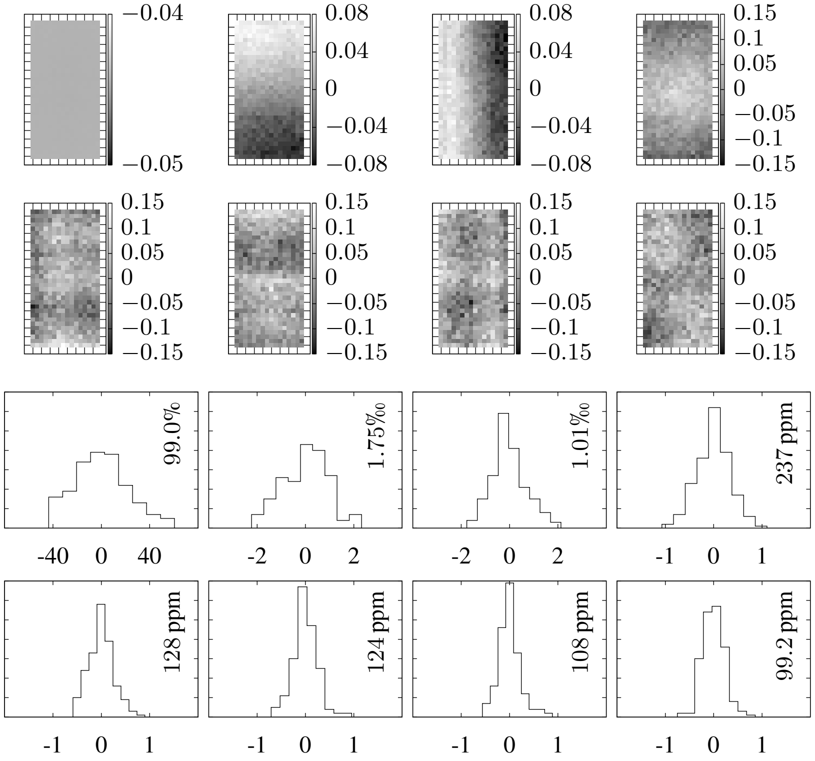
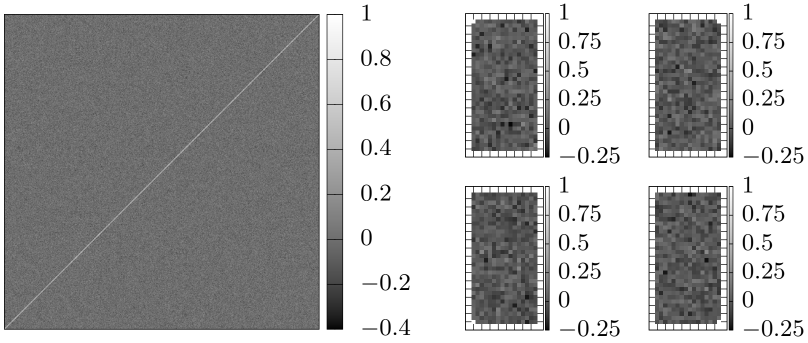
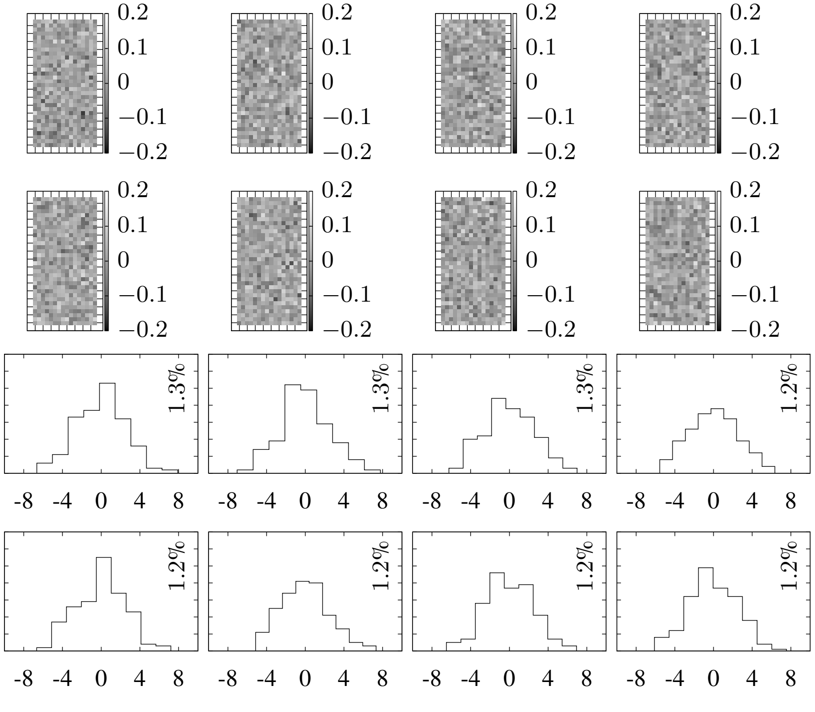


Figure 4.5.: First eight principal components (heatmaps), histograms of estimations, and explained ratio of variance (printed value) for RO frequencies (mean among accesses) in Maiti et al. dataset. Reproduced from [**71**].

The size of the dataset is equal to that of the RO frequency data by Maiti et al. to rule out differences induced from dataset size. As expected, the correlation matrix in Fig. 4.6a varies around zero for all elements except the main diagonal, which by definition has a value of one, and no identifiable pattern exists if selected columns are plotted in a pseudo-layout preserving way. Accordingly, a PCA shows that all principal components describe about the same amount of variance, see Fig. 4.6b, because there exists no linear transformation that would allow to describe more than a single apparent RV's contribution by any transformed RV. This provides confidence in the results for the ROmaiti dataset to actually have found spatial correlation and not being coincidental.



(a) Correlation matrix and selected elements in layout preserving form



(b) First eight principal components (heatmaps), histograms of estimations, and explained ratio of variance (printed value)

Figure 4.6.: Correlation matrix and PCA results for artificial independent standard normal distributed data

Crafted Binary Response Data

In the previous examples, PCA seemed well suited to detect varying gradients that cause distributed correlation on continuous response data such as the RO frequencies by Maiti et al. To show that PCA can also be applied on binary data, the following example uses a specifically crafted dataset that contains gradient effects of varying direction and slope similar to the frequencies in the ROmaiti dataset, but in binary form. It is created as follows: For each virtual device, a row slope δ_d and a column slope ϵ_d is tossed from a uniform distribution in the interval $[0, 1)$, then linearly spaced vectors $\underline{\delta}_d = (\delta_d, \ldots, 1 - \delta_d)$, $\underline{\epsilon}_d = (\epsilon_d, \ldots, 1 - \epsilon_d)$ are created, which contain as many elements as there are columns respectively rows in the layout. These vectors are then combined into the success probability

$$p_{b,d} = \frac{\delta_{b'',d} + \epsilon_{b',d}}{2}, \tag{4.20}$$

where $b \in \{1, \ldots, B\}$ and

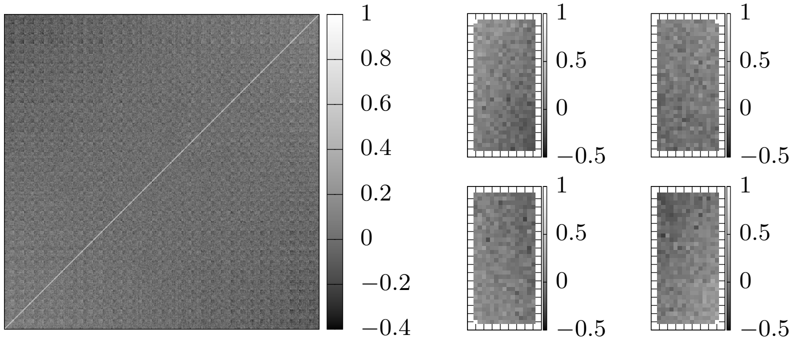
$$b' = \left\lceil \frac{b}{B''} \right\rceil \quad \in \{1, \ldots, B'\} \tag{4.21}$$

iterates over the B' rows and

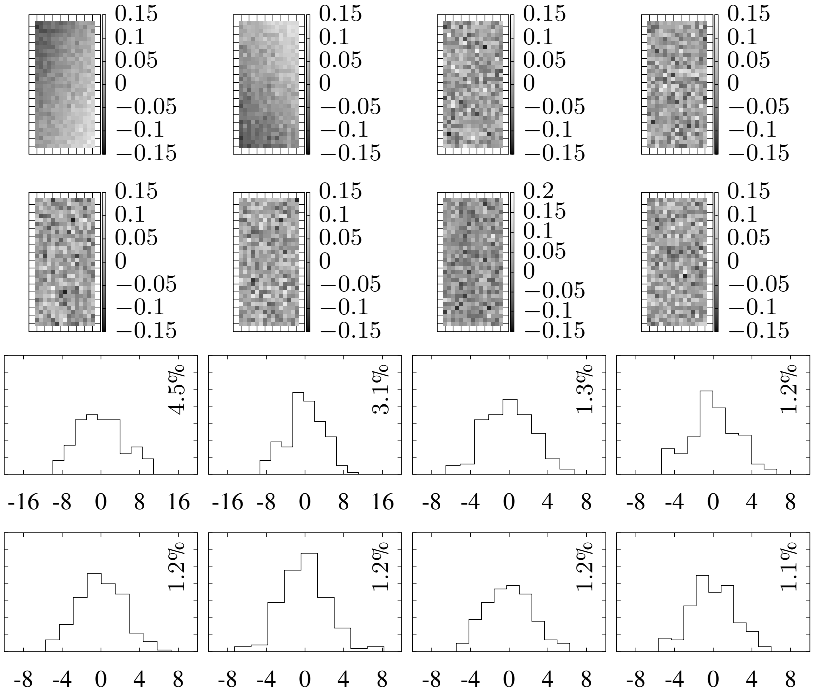
$$b'' = ((b-1) \bmod B'') + 1 \quad \in \{1, \ldots, B''\} \tag{4.22}$$

iterates over the B'' columns. So the success probability, if plotted in a layout preserving way, is a plane tilted according to δ_d and ϵ_d. For each bit of response in the dataset, a single Bernoulli trial with the corresponding success probability is made to finally obtain the actual dataset. The size of the dataset is chosen equal to what would result from an overlapping pairwise comparison of ROs. This results in $B' = 32$ rows with $B'' = 15$ columns and $D = 193$ devices.

The correlation matrix in Fig. 4.7a shows a texture similar to that in Fig. 4.4b, though less pronounced. Still, the PCA is able to identify the gradient, see Fig. 4.7b, as the first two principal components represent 4.5% respectively 3.1% of variation in the dataset, while the third and following principal components each explain at most 1.3% of variation. Although the first two principal components do not map to the x and y axis, but to the down-right respectively the up-right diagonal, this is still an orthonormal basis for the tilted plain that is the induced flaw, just less intuitive. This shows that PCA does work on binary response data, too, as long as the flaw is not removed by the bit extraction algorithm.



(a) Correlation matrix and selected elements in layout preserving form



(b) First eight principal components (heatmaps), histograms of estimations, and explained ratio of variance (printed value)

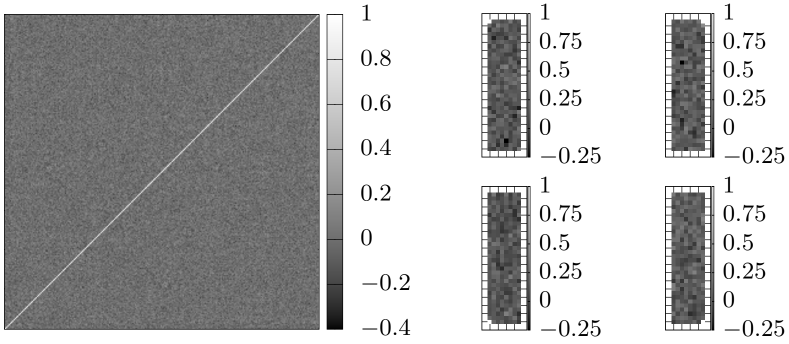
Figure 4.7.: Correlation matrix and PCA results for artificial binary dataset with varying gradient

Binary Data Based on Maiti et al.

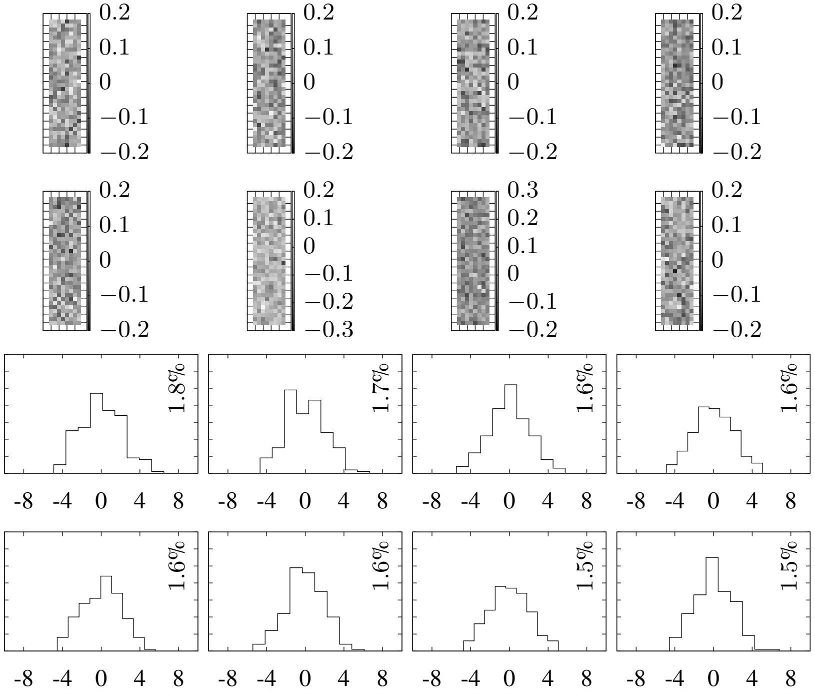
Now as the applicability of PCA on binary data has been shown, the effect of bit extraction on the consequences of spatial artifacts can be investigated. Therefore a PCA is applied now on binary responses from the ROmaiti dataset obtained by MEPWC and overlapping pairwise comparison.

In the first case, bit extraction is performed as in [**71**] by MEPWC of horizontally adjacent ROs, cf. (1.13). This allows to conveniently locate the extracted bits at the shared border of the compared ROs, resulting in an 8 by 32 on-die layout. As mentioned above, the MEPWC acts as a spatial high-pass filter, so the impact of the spatial artifact on the binary response is strongly attenuated. This is recognizable in the correlation matrix in Fig. 4.8a and the PCA results in Fig. 4.8b, since both are close to the plots for the independent standard normal distributed data and the explained variance among the principal components decreases smoothly. If the gradients within the frequency data would still be present, they would cause the first few components to explain a much larger portion of variance than the following principle components.

In the second case, bit extraction is performed in an overlapping way without comparison across lines, cf. (1.12). This provides an array of 15 by 32 response bits, which contains almost twice as many bits as with the previous approach. It comes, based on the discussion in Sec. 1.5.1, at the price of correlations among the response bits, though. Yet, from the PCA results in Fig. 4.9b it seems there are no such correlations. By close inspection of the correlation matrix in Fig. 4.9a, one might notice that the first subdiagonals are darker, which is also recognizable in the selected columns plotted on the right hand side. However, since the correlation is hard to spot and seems to affect only a very small part of the correlation matrix, the severity of the issue is likely to be underestimated or even completely overlooked. This is an example where SPACA shines, because it raises an unambiguous red flag on the data, as detailed in Section 4.2.2.

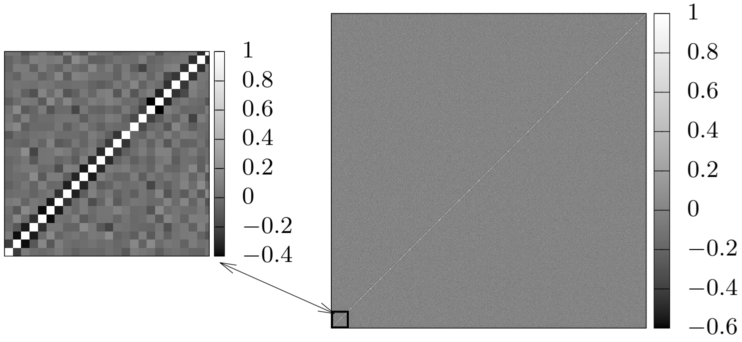


(a) Correlation matrix and selected elements in layout preserving form

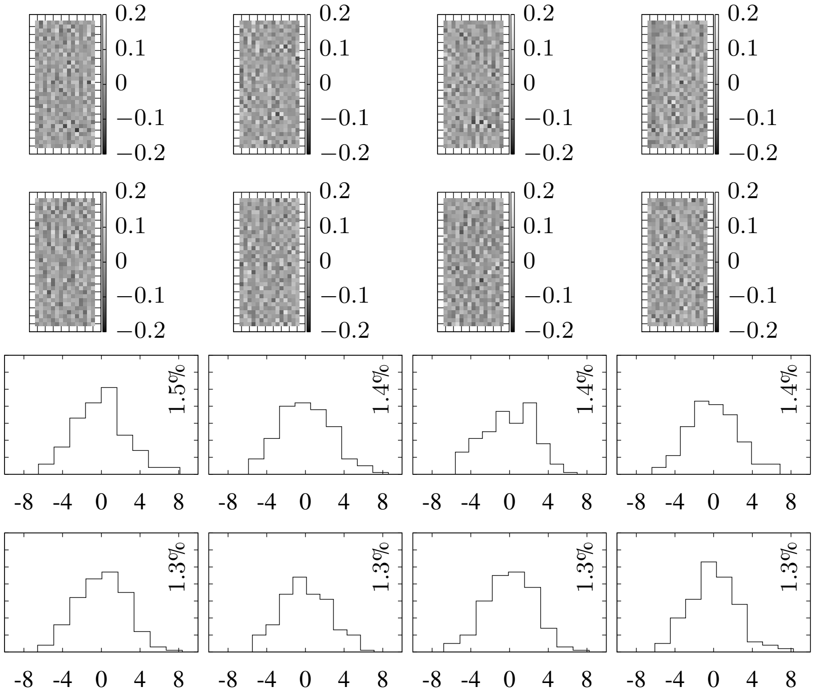


(b) First eight principal components (heatmaps), histograms of estimations, and explained ratio of variance (printed value)

Figure 4.8.: Correlation matrix and PCA results for binary responses extracted by MEPWC from ROmaiti dataset



(a) Correlation matrix with detailed view to show darker first subdiagonals



(b) First eight principal components (heatmaps), histograms of estimations, and explained ratio of variance (printed value)

Figure 4.9.: Correlation matrix and PCA results for binary responses extracted by overlapping comparison from ROmaiti dataset

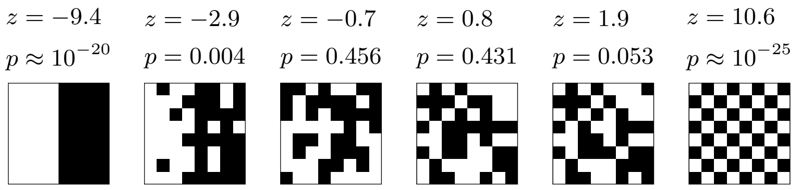


Figure 4.10.: Crafted examples of spatial autocorrelation in an 8 by 8 binary matrix together with z-scores and p-values of $\overset{\bullet\circ}{J}$ under the non-free sampling assumption and a rook's neighborhood. See Tbl. 4.4 for corresponding formulae.

4.2.2. Spatial Autocorrelation Analysis

As briefly mentioned in the beginning of Sec. 4.2, spatial autocorrelation may arise in a negative or positive way. The difference is intuitively understood by considering their most extreme form in a binary two dimensional setting such as a wall with black and white tiles. In the case of perfect positive spatial autocorrelation, white tiles could be separated from black tiles by a single line, so e.g. all white tiles on the left hand side and all black tiles on the right hand side. Under perfect negative spatial autocorrelation, the wall would resemble a chess board, where black and white tiles perfectly interleave in every direction. More general, positive spatial autocorrelation is related to a clustering of similar values, where negative spatial autocorrelation relates to dispersion. Examples with varying extent of spatial autocorrelation can be found in Fig. 4.10. While spatial autocorrelation seems obvious at such high degrees, a PUF's unpredictability can be affected already at much smaller degrees of spatial autocorrelation, where a human eye may not yet spot it. It is therefore reasonable to use a specialized tool such as SPACA to test for it. A preliminary version of this subsection has been published in [**60**].

SPACA originates from the field of statistics at around the 1950s and had its first applications on agricultural [103] and morbidity [104] data. Well known methods among statisticians are Moran's I [103], Geary's c [104], and the Join Count statistic J [105], which have been improved and extended over time by several authors, see [**60**] for details. The three methods have many things in common, e.g. they all conduct a statistical hypothesis test with the null hypothesis H_0 that there is no spatial autocorrelation in the data, iterate over the data by comparing samples from neighbouring locations, and their output is asymptotically normal distributed. Their main difference lies in the type of data, where Moran's I and Geary's c are for continuously valued data and the Join Count statistic J is for discrete, typically

Sums Over Adjacency Weights

$$S_0 = \sum_{b=1}^{B} \sum_{b'=1}^{B} \omega_{b,b'} \tag{4.23}$$

$$S_1 = \frac{1}{2} \sum_{b=1}^{B} \sum_{b'=1}^{B} \left(\omega_{b,b'} + \omega_{b',b}\right)^2 \tag{4.24}$$

$$S_2 = \sum_{b=1}^{B} \left(\sum_{b'=1}^{B} \left(\omega_{b,b'} + \omega_{b',b}\right) \right)^2 \tag{4.25}$$

Central and Standardized Sample Moments (substitute $\bar{x}_{b,d}$ for $\bar{\xi}_{b,d}$ if necessary)

$$m_d = \frac{1}{B} \sum_{b=1}^{B} \bar{\xi}_{b,d} \qquad \text{(mean)} \tag{4.26}$$

$$k_d = B \frac{\sum_{b=1}^{B} \left(\bar{\xi}_{b,d} - m_d\right)^4}{\left(\sum_{b=1}^{B} \left(\bar{\xi}_{b,d} - m_d\right)^2\right)^2} \qquad \text{(kurtosis)} \tag{4.27}$$

Table 4.1.: Shorthand notations used in formulae of SPACA

binary valued, data, although extensions to more than two classes exist, see [106]. Test statistic and expected moments under H_0 for each method may be found in Tbl. 4.2 for Moran's I, in Tbl. 4.3 for Geary's c, and in Tbl. 4.4 for the Join Count statistic. They use a common list of shorthands found in Tbl. 4.1.

To conduct SPACA, several steps are necessary regardless of the chosen method:

1. Choose a significance level α for the hypothesis test and an adjacency matrix $\underline{\underline{\omega}}$, also known as connection or weighting matrix, that defines which locations are deemed neighbors and how proximate these are. This step needs to be done only once, whereas the following steps need to be done per device.

2. Calculate expectation and variance of the test statistic under H_0, given the adjacency list and some empirical moments or sample counts.

3. Calculate the test statistic on the measured data.

4. Decide upon H_0 per device based on the p-value that corresponds to the results of the previous two steps.

The steps are now explained in more detail, where a focus on the application to PUFS is made. Examples on several real-world and crafted datasets follow to display the worthwhile insights that this kind of analysis provides.

Test Statistics and Expected Moments

The test statistic for Moran's I (4.28) has a value range of -1 to 1, where -1 indicates strong negative spatial autocorrelation, +1 reveals strong positive spatial autocorrelation, and a value close to zero is expected if the observations are not spatially correlated. The test statistic for Geary's c (4.35) instead has a value range of 0 to 2, where positive spatial autocorrelation leads to a result closer to 0, and negative spatial autocorrelation increases the result towards 2. A result close to unity is expected if the observations are not spatially correlated. For both methods, the expectation for spatially uncorrelated observations is independent of the chosen adjacency matrix and the distribution the observations are drawn from. It is further independent of the number of analyzed observations for Geary's c, but depends on it for Moran's I. The expected variance of the test statistics under H_0 depends foremost on the number of observations and the adjacency matrix, because it influences how many comparisons are made and the more comparisons, the stronger the amplitude for the same amount of spatial autocorrelation. In addition, it depends on the kurtosis k_d respectively success probability $\hat{p}_d$ of the probability distribution of the observations if they are not independent drawings from the same normal distribution. Then, the so-called randomization approach allows to calculate the expected variance of the test statistic for arbitrary probability distributions, as long as the observations are all drawn from the same distribution. In such case, the variance of the test statistic is estimated relative to all possible permutations of the made observations to the available locations.

For the join count statistic, one may test for spatial autocorrelation from either black to black, white to white, or black to white joins, denoted as $\overset{\bullet\bullet}{J}$, $\overset{\circ\circ}{J}$, and $\overset{\bullet\circ}{J}$, respectively. Black indicates in this case the presence of a feature or a response bit being 1, and white refers to the absence of a feature or a response bit being 0. In logical notation, the comparison of joins corresponds to AND for black to black, NOR for white to white, and XOR for black to white. In mathematical notation, with 1 and 0 representing the presence and absence of a feature, the XOR in (4.40) coincides with the comparison function of Geary's c, i.e. $(\bar{x}_{b,d} - \bar{x}_{b',d})^2$. If the observations are positively spatial autocorrelated, the number of black to white joins decreases and the number of other joins increases, because adjacent locations tend to produce the same response. If the observations are negatively

Test Statistic

$$I_d = \frac{B}{S_0} \frac{\sum_{b=1}^{B-1} \sum_{b'=b+1}^{B} (\omega_{b,b'} + \omega_{b',b})(\bar{\xi}_{b,d} - m_d)(\bar{\xi}_{b',d} - m_d)}{\sum_{b=1}^{B} \left(\bar{\xi}_{b,d} - m_d\right)^2} \tag{4.28}$$

Expected Moments Under Normality Assumption

$$\underset{\mathrm{N}}{\mathrm{E}}[I] = \frac{-1}{B-1} \tag{4.29}$$

$$\underset{\mathrm{N}}{\mathrm{E}}[I^2] = \frac{B^2 S_1 - B S_2 + 3S_0^2}{(B^2-1)S_0^2} \tag{4.30}$$

$$\underset{\mathrm{N}}{\mathrm{Var}}[I] = \underset{\mathrm{N}}{\mathrm{E}}[I^2] - \underset{\mathrm{N}}{\mathrm{E}}[I]^2 \tag{4.31}$$

Expected Moments Under Randomization Assumption

$$\underset{\mathrm{R}}{\mathrm{E}}[I] = \frac{-1}{B-1} \tag{4.32}$$

$$\underset{\mathrm{R}}{\mathrm{E}}[I_d^2] = \frac{\begin{pmatrix} B\left(\left(B^2 - 3B + 3\right) S_1 - BS_2 + 3S_0^2\right) \\ -k_d\left(\left(B^2 - B\right) S_1 - 2BS_2 + 6S_0^2\right) \end{pmatrix}}{(B-1)(B-2)(B-3)S_0^2} \tag{4.33}$$

$$\underset{\mathrm{R}}{\mathrm{Var}}[I_d] = \underset{\mathrm{R}}{\mathrm{E}}[I_d^2] - \underset{\mathrm{R}}{\mathrm{E}}[I]^2 \tag{4.34}$$

Table 4.2.: Test statistic and expected moments under H_0 for Moran's I.

Test Statistic

$$c_d = \frac{B-1}{2S_0} \frac{\sum_{b=1}^{B-1} \sum_{b'=b+1}^{B} (\omega_{b,b'} + \omega_{b',b})(\bar{\xi}_{b,d} - \bar{\xi}_{b',d})^2}{\sum_{b=1}^{B} \left(\bar{\xi}_{b,d} - m_d\right)^2} \tag{4.35}$$

Expected Moments Under Normality Assumption

$$\underset{\mathrm{N}}{\mathrm{E}}[c] = 1 \tag{4.36}$$

$$\underset{\mathrm{N}}{\mathrm{Var}}[c] = \frac{(B-1)(2S_1 + S_2) - 4S_0^2}{2(B+1)S_0^2} \tag{4.37}$$

Expected Moments Under Randomization Assumption

$$\underset{\mathrm{R}}{\mathrm{E}}[c] = 1 \tag{4.38}$$

$$\underset{\mathrm{R}}{\mathrm{Var}}[c_d] = \frac{\begin{pmatrix} \left(B^2 - 3 - (B-1)^2 k_d\right) S_0^2 \\ +(B-1)\left(B^2 - 3B + 3 - (B-1)k_d\right) S_1 \\ -\frac{1}{4}(B-1)\left(B^2 + 3B - 6 - (B^2 - B + 2)k_d\right) S_2 \end{pmatrix}}{B(B-2)(B-3)S_0^2} \tag{4.39}$$

Table 4.3.: Test statistic and expected moments under H_0 for Geary's c.

Test Statistic

$$\overset{\bullet\circ}{J}_d = \frac{1}{2}\sum_{b=1}^{B-1}\sum_{b'=b+1}^{B}(\omega_{b,b'} + \omega_{b',b})(\bar{x}_{b,d} \oplus \bar{x}_{b',d}) \tag{4.40}$$

Expected Moments Under Free Sampling Assumption

$$\underset{\mathrm{F}}{\mathrm{E}}\left[\overset{\bullet\circ}{J}_d\right] = S_0\hat{p}_d(1-\hat{p}_d) \tag{4.41}$$

$$\underset{\mathrm{F}}{\mathrm{Var}}\left[\overset{\bullet\circ}{J}_d\right] = \frac{1}{4}S_2\hat{p}_d(1-\hat{p}_d) + (S_1 - S_2)\hat{p}_d^2(1-\hat{p}_d)^2 \tag{4.42}$$

$$\text{with} \quad \hat{p}_d = \mathrm{P}(\bar{x}_{b,d} = 1)$$

Expected Moments Under Non-Free Sampling Assumption

$$\underset{\mathrm{NF}}{\mathrm{E}}\left[\overset{\bullet\circ}{J}_d\right] = \frac{1}{2}S_0\frac{\overset{\bullet}{n}_d\overset{\circ}{n}_d}{\binom{B}{2}} \tag{4.43}$$

$$\underset{\mathrm{NF}}{\mathrm{E}}\left[\overset{\bullet\circ}{J}{}_d^{2}\right] = \frac{1}{4}\left(\begin{array}{c} S_1\dfrac{\overset{\bullet}{n}\overset{\circ}{n}}{\binom{B}{2}} + (S_2 - 2S_1)\dfrac{\overset{\bullet}{n}(\overset{\bullet}{n}+\overset{\circ}{n}-2)\overset{\circ}{n}}{B(B-1)(B-2)} \\ +4(S_0^2 + S_1 - S_2)\dfrac{\overset{\bullet}{n}(\overset{\bullet}{n}-1)\overset{\circ}{n}(\overset{\circ}{n}-1)}{B(B-1)(B-2)(B-3)} \end{array}\right) \tag{4.44}$$

$$\underset{\mathrm{NF}}{\mathrm{Var}}\left[\overset{\bullet\circ}{J}_d\right] = \underset{\mathrm{NF}}{\mathrm{E}}\left[\overset{\bullet\circ}{J}{}_d^{2}\right] - \underset{\mathrm{NF}}{\mathrm{E}}\left[\overset{\bullet\circ}{J}_d\right]^2 \tag{4.45}$$

$$\text{with} \quad \overset{\bullet}{n}_d = \sum_{b=1}^{B}\bar{x}_{b,d} \quad \text{and} \quad \overset{\circ}{n}_d = B - \overset{\bullet}{n}_d$$

Table 4.4.: Test statistic and expected moments under H_0 for the Join Count statistic of black to white joins. Those for black to black and white to white joins may be found in [**60**].

spatial autocorrelated, the number of black to white joins increases and the number of other joins decreases, because adjacent locations tend to produce different responses. For clarity, only the black to white join count is used in the following, for other types of joins see [**60**]. Similar to Moran's I and Geary's c, there exist two approaches to determine the expectation and expected variance under H_0: the free sampling and the non-free sampling approach. If the observations are independent drawings from the same Bernoulli distribution with success parameter $\hat{p}$, but without a fixed amount of successes and failures, the moments may be calculated under the free sampling approach. If instead the number of successes and failures is fixed and the test statistic shall be evaluated relative to all possible permutations of the successes on the locations, the non-free sampling assumption is to be applied.

Adjacency Matrix

The adjacency matrix $\underline{\underline{\omega}}$ with elements $\omega_{b,b'}$ provides for each pair of locations b, b' a weight that relates to the proximity of the locations in some distance metric. Note that, as stated in the beginning of this section, a bijective relationship between a location on the die and index b is assumed, e.g. because each PUF cell produces exactly one response bit or b is redefined to index multi-bit symbols. Depending on what kind of correlation is suspected and should be tested for, this can be physical distance on the die, electrical distance with regard to the supply grid, etc. In both cases, the matrix will be symmetric, since the physical distance of location b to b' is the same as vice versa and in an ohmic supply line with two taps the partial derivatives are identical in both directions:

$$U_1 = U_0 - R_1(I_1 + I_2) \qquad U_2 = U_0 - R_1(I_1 + I_2) - R_2 I_2 \tag{4.46}$$

$$\frac{\partial U_1}{\partial I_2} = -R_1 \qquad \frac{\partial U_2}{\partial I_1} = -R_1 \tag{4.47}$$

However, asymmetric adjacency matrices are also supported by SPACA without any overhead, because all three methods use commutative operations to compare the data: $(\bar{\xi}_{b,d} - m_d)(\bar{\xi}_{b',d} - m_d)$ for Moran's I and $(\bar{x}_{b,d} - \bar{x}_{b',d})^2$ for Geary's c and $\overset{\bullet\circ}{J}$.

Regardless of symmetry, fully populated adjacency matrices may cause prohibitive computational costs, since they grow quadratically with the number of locations and so does the number of comparisons required to calculate the test statistic. The common solution to this is to define a rather small neighborhood range and set the adjacency for all other pairs to zero, so they can be skipped during calculation of the test statistic. The assumption for this approach is that spatial dependencies are contiguous, so they may affect direct neighbors or larger areas, but not spatially distant locations without any of those in between.

Common neighborhoods are rook's move (upper, lower, left, and right neighbor with unity weight), queen's move (with additional upper-left, upper-right, lower-left, and lower-right neighbor with unity weight), or k-nearest neighbor with weights indirect proportional to distance. Fig. 4.11 shows a rook's neighborhood together with the corresponding adjacency matrix for a 4 by 4 location array. The first upper and lower subdiagonal contain the left and right neighbors if locations are indexed from top left to the right, and row by row. In these first subdiagonals, three 1s followed by a 0 arise from the fact that the leftmost location in every row of the array does not have a left neighbor and the rightmost location does not have a right neighbor. The 1s in the fourth upper and lower subdiagonal represent the lower and upper neighbors, respectively.

As will be shown in the following examples with real-world datasets, a rook's move neighborhood is sufficient to detect spatial autocorrelation in a PUF candidate most of the time. To verify this, though, larger neighborhoods have to be evaluated for comparison. Therefore rook's move is generalized herein as a taxicab norm with distance limit. The taxicab norm measures distance in rectangular movements, i.e. the upper-left neighbor of a location has distance two, because it requires on step to the left and one step upwards, or vice versa. Diagonal moves are not allowed in the taxicab norm. From this point of view, a rook's move neighborhood equals a taxicab neighborhood with distance limit one. A larger neighborhood would then be e.g. a distance limit of three, cf. Fig. 4.12.

In particular cases, an irregular taxicab norm, where horizontal and vertical steps have different weight, is required to avoid spatial autocorrelation in different directions to cancel each other out. Two examples of such irregular taxicab norms with distance limit 3 and ratio $1/2$ and 2 are shown in Fig. 4.13. They will be used later in this section in the analysis of the SRAMxmc dataset.

Decision Making

In a typical statistical hypothesis tests, H_0 is to be rejected if the so-called p-value, i.e. the probability to observe this or a more extreme value of the test statistic if H_0 would hold, is below the significance level α. Since the test statistic for all three methods exhibit asymptotic normality under H_0, a reasonable way to calculate the p-value is via so-called z-scores, i.e. by translation to a standard normal distribution. For Moran's I under normality assumption and randomization approach this means

$$\overset{\langle \mathrm{N} \rangle}{I_d} = \frac{I_d - \mathrm{E}_\mathrm{N}[I]}{\sqrt{\mathrm{Var}_\mathrm{N}[I]}} \quad \text{respectively} \quad \overset{\langle \mathrm{R} \rangle}{I_d} = \frac{I_d - \mathrm{E}_\mathrm{R}[I]}{\sqrt{\mathrm{Var}_\mathrm{R}[I]}}. \tag{4.48}$$

$$\begin{tikzcd}
0_1 & 0_2 & 0_3 & 0_4 \\
0_5 & 0_6 & 1_7 & 0_8 \\
0_9 & 1_{10} & \cdot \arrow[l] \arrow[r] \arrow[u] \arrow[d] & 1_{12} \\
0_{13} & 0_{14} & 1_{15} & 0_{16}
\end{tikzcd}$$

(a) Rook's move includes upper, lower, left, and right neighbor. Subscripts are location indices.

$$\begin{array}{r}
1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \\ 12 \\ 13 \\ 14 \\ 15 \\ 16
\end{array}
\begin{pmatrix}
0&1&0&0&1&0&0&0&0&0&0&0&0&0&0&0\\
1&0&1&0&0&1&0&0&0&0&0&0&0&0&0&0\\
0&1&0&1&0&0&1&0&0&0&0&0&0&0&0&0\\
0&0&1&0&0&0&0&1&0&0&0&0&0&0&0&0\\
1&0&0&0&0&1&0&0&1&0&0&0&0&0&0&0\\
0&1&0&0&1&0&1&0&0&1&0&0&0&0&0&0\\
0&0&1&0&0&1&0&1&0&0&1&0&0&0&0&0\\
0&0&0&1&0&0&1&0&0&0&0&1&0&0&0&0\\
0&0&0&0&1&0&0&0&0&1&0&0&1&0&0&0\\
0&0&0&0&0&1&0&0&1&0&1&0&0&1&0&0\\
0&0&0&0&0&0&1&0&0&1&0&1&0&0&1&0\\
0&0&0&0&0&0&0&1&0&0&1&0&0&0&0&1\\
0&0&0&0&0&0&0&0&1&0&0&0&0&1&0&0\\
0&0&0&0&0&0&0&0&0&1&0&0&1&0&1&0\\
0&0&0&0&0&0&0&0&0&0&1&0&0&1&0&1\\
0&0&0&0&0&0&0&0&0&0&0&1&0&0&1&0
\end{pmatrix}$$

(b) Adjacency matrix for rook's move in a 4 by 4 array with location index on the left.

Figure 4.11.: Rook's move neighborhood and corresponding adjacency matrix for a 4 by 4 array.

$$\begin{tikzcd}
0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 3 & 2 & 3 & 0 & 0 & 0 \\
0 & 0 & 3 & 2 & 1 & 2 & 3 & 0 & 0 \\
0 & 3 & 2 & 1 & \cdot \arrow[l] \arrow[r] \arrow[u] \arrow[d] & 1 & 2 & 3 & 0 \\
0 & 0 & 3 & 2 & 1 & 2 & 3 & 0 & 0 \\
0 & 0 & 0 & 3 & 2 & 3 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0
\end{tikzcd}$$

Figure 4.12.: A taxicab norm neighborhood with distance limit 3. Note that the weights in $\underline{\underline{\omega}}$ are the reciprocal of the distance shown here.

$$\begin{array}{ccccccc}
0 & 0 & 0 & 3.0 & 0 & 0 & 0 \\
0 & 0 & 0 & 2.5 & 0 & 0 & 0 \\
0 & 0 & 3.0 & 2.0 & 3.0 & 0 & 0 \\
0 & 0 & 2.5 & 1.5 & 2.5 & 0 & 0 \\
0 & 3.0 & 2.0 & 1.0 & 2.0 & 3.0 & 0 \\
0 & 2.5 & 1.5 & 0.5 & 1.5 & 2.5 & 0 \\
3.0 & 2.0 & 1.0 & \leftarrow \overset{\uparrow}{\underset{\downarrow}{\cdot}} \rightarrow & 1.0 & 2.0 & 3.0 \\
0 & 2.5 & 1.5 & 0.5 & 1.5 & 2.5 & 0 \\
0 & 3.0 & 2.0 & 1.0 & 2.0 & 3.0 & 0 \\
0 & 0 & 2.5 & 1.5 & 2.5 & 0 & 0 \\
0 & 0 & 3.0 & 2.0 & 3.0 & 0 & 0 \\
0 & 0 & 0 & 2.5 & 0 & 0 & 0 \\
0 & 0 & 0 & 3.0 & 0 & 0 & 0
\end{array}$$

(a) Ratio $^1/_2$

$$\begin{array}{ccccccc}
0 & 0 & 3.0 & 2.0 & 3.0 & 0 & 0 \\
3.0 & 2.0 & 1.0 & \leftarrow \overset{\uparrow}{\underset{\downarrow}{\cdot}} \rightarrow & 1.0 & 2.0 & 3.0 \\
0 & 0 & 3.0 & 2.0 & 3.0 & 0 & 0
\end{array}$$

(b) Ratio 2

Figure 4.13.: Irregular taxicab norm with distance limit 3 and ratio $^1/_2$ and 2 to avoid that spatial autocorrelation in different directions cancels each other out. Compared to a regular taxicab norm, blocks are no longer quadratic but just rectangular, so steps in different directions add a different amount to the overall distance. Note that the weights in $\underline{\underline{\omega}}$ are the reciprocal of the distance shown here.

Application to Geary's c and the join count statistic is identical. The p-value is then obtained from the z-score by use of the complementary error function, e.g. for Moran's I under normality assumption

$$p\left(\overset{\langle \mathrm{N} \rangle}{I_d}\right) = \operatorname{erfc}\left(\frac{\left|\overset{\langle \mathrm{N} \rangle}{I_d}\right|}{\sqrt{2}}\right). \tag{4.49}$$

Note that although this is a two-sided test, H_0 is to be rejected if the p-value is smaller than or equal to α, because the complementary error function already yields the two-sided probability that a $\mathfrak{N}(0, 1/2)$ distributed RV lies outside the interval $[-t, t]$ for positive t. If a larger set of devices needs to be tested, it is more efficient to calculate limits for the z-scores or the test statistic rather than to calculate the p-value for each device.

In addition to single device tests, a representative group of devices may be analyzed to evaluate the design of a PUF candidate e.g. during a qualification test. Because the z-scores approach a standard normal distribution under H_0 as mentioned earlier, the design and production process rather than an individual device may be evaluated based on the distribution of the z-scores of such a representative group. Even if all devices pass individually, flaws in the design or production process are suggested if the z-scores do not follow a standard normal distribution.

Optimized Implementation for Large Regular Arrays

While implementations for the methods described above are readily available e.g. from the Python spatial analysis library (PySAL), they follow a general purpose approach and are not optimized towards PUFs. Therefore, an application specific implementation in MATLAB R2016b is provided [**107**] that is optimized with respect to performance by two simplifications: First, locations are in a regular two-dimensional array and thus can be represented in matrix form. Second, the neighborhood is small compared to the size of the array, and it is symmetric and stationary. To understand why these assumptions improve performance, it is necessary to note that vectorized calculations and built-in functions are typically much faster than iteration in user level scripts. Without above assumptions, a straight forward implementation of the methods would require two nested loops in a user script that iterate through all pairs of locations, see (4.28) for example, and thus be of complexity $\mathcal{O}(B^2)$, although many of the loop iterations would not contribute to the result since the corresponding entry of the adjacency matrix is zero. By utilizing these assumptions, though, the user script can iterate over the $m(\ll B)$ neighbors in the neighborhood rather than all B locations and no user-level loop iterations need to be spent to detect zeros in the weighting matrix. So instead of calculating all neighbors for a particular location, a particular neighbor such as the left or upper-right neighbor is calculated for all locations, where iteration over the B locations is done by fast vectorized operations and built-in functions. Since the number of neighbors is assumed much smaller than and independent of the number of locations, complexity is reduced to $\mathcal{O}(B)$. As an example for these optimizations, the nominator of Geary's c for a rook's move neighborhood requires only two user script level iterations. One for the left and right neighbor

$$2\texttt{sum}\left(\begin{pmatrix} \bar{\xi}_{1,2} & \cdots & \bar{\xi}_{1,B''} \\ \vdots & \ddots & \vdots \\ \bar{\xi}_{B',2} & \cdots & \bar{\xi}_{B',B''} \end{pmatrix} - \begin{pmatrix} \bar{\xi}_{1,1} & \cdots & \bar{\xi}_{1,B''-1} \\ \vdots & \ddots & \vdots \\ \bar{\xi}_{B',1} & \cdots & \bar{\xi}_{B',B''-1} \end{pmatrix}\right)^2,$$

and one for the upper and lower neighbor

$$2\texttt{sum}\left(\begin{pmatrix} \bar{\xi}_{2,1} & \cdots & \bar{\xi}_{2,B''} \\ \vdots & \ddots & \vdots \\ \bar{\xi}_{B',1} & \cdots & \bar{\xi}_{B',B''} \end{pmatrix} - \begin{pmatrix} \bar{\xi}_{1,1} & \cdots & \bar{\xi}_{1,B''} \\ \vdots & \ddots & \vdots \\ \bar{\xi}_{B'-1,1} & \cdots & \bar{\xi}_{B'-1,B''} \end{pmatrix}\right)^2,$$

where in both cases the square is taken element-wise and `sum` is a built-in operation that yields the sum of all elements of the matrix. For a queen's move neighborhood, two additional iterations would be necessary, one for the upper-left and lower-right neighbor, and one for the upper-right and lower-left neighbor. While the available implementation is currently limited to symmetric stationary neighborhoods, it could be extended to asymmetric stationary neighborhoods if the outermost factor $2\omega_{b,b'}$ is replaced by $\omega_{b,b'} + \omega_{b',b}$. The benefit of this approach is most obvious for large B such as for the SRAMxmc dataset, where $B = 1\,310\,720$.

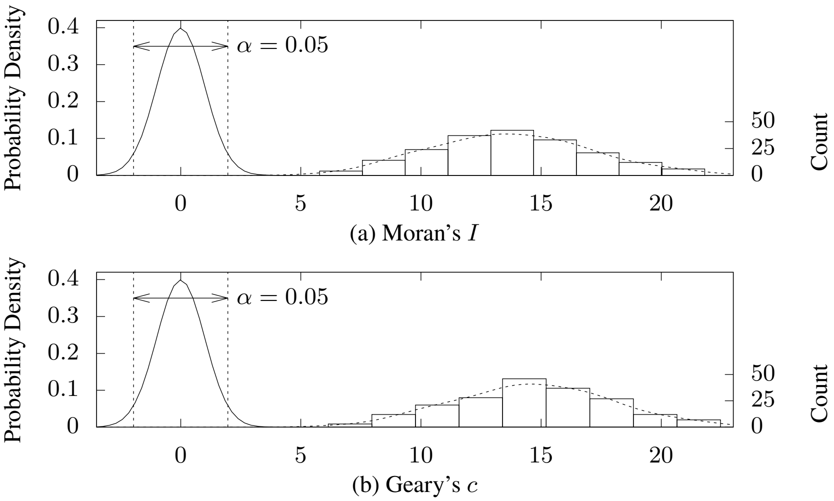
Examples With Real-World Datasets

To exemplify the usefulness of SPACA for PUF testing, it is applied on the ROmaiti, SRAMsu, and SRAMxmc dataset in the following. Some of these examples appeared previously in [**60**]. Expected moments for H_0 are calculated based on the randomization approach and correspondingly the non-free sampling approach for the join count statistic. Resulting z-scores are visualized in histograms together with a kernel density estimation (KDE) for easier comparison to the overlaid standard normal distribution (SND), which is the expectation under H_0. Bins are constructed using Scott's rule to avoid overfitting. Vertical bars indicate the acceptance interval for $\alpha = 0.05$. The two y-axes for probability density (left) and event count (right) are scaled so that the area covered by the histogram bars matches the area under the SND.

ROmaiti Application of Moran's I and Geary's c with a rook's move neighborhood on the true RO frequencies $\bar{\xi}_{b,d}$ of the ROmaiti dataset provides z-scores entirely separated to the right of the SND, which indicates strong positive spatial autocorrelation, see Fig. 4.14. Besides the speed gradients already found through PCA, this result is partially also caused by the expected frequency of a RO to vary based on its location, which is a known defect of the dataset shown in Fig. 1.12. Because SPACA tests consider each device individually, they cannot distinguish whether a certain pattern on a device is due to unequal expectation or true spatial autocorrelation. This cross-sensitivity is not obstructive, though, because unequal expected frequencies cause bias and thus also impair the unpredictability of the response. So if SPACA flags a device, the question is rather on the exact reason, but not whether the device is free of flaws. If the mean frequencies differ sufficiently from one position to the other, most devices will also be flagged regarding spatial autocorrelation. If mean frequencies are equal for all positions, true spatial autocorrelation is causative. To determine the actual reason, SPACA can be combined with appropriate bias tests or by repetition of SPACA once mean frequency is equalized. The latter approach leads to Fig. 4.15, which repeats the same analysis on $\bar{\xi}_{b,d} - m_b$ with m_b from (4.26), so the expected frequency is equalized among all locations. Although the results moved noticeably closer to the SND, virtually all devices are still outside the acceptance region for an $\alpha = 0.05$ significance level, so one may conclude that the ROmaiti dataset suffers from both unequal expectation and true spatial autocorrelation.

Comparing these results to those found by PCA, two aspects are noteworthy: The unequal expected frequency cannot be detected by PCA, as it operates on z-scores normalized along the dimension of devices. However, this may be noticed from a mere plot of expectation and standard deviation as in Fig. 1.12. The speed gradients over the die are flagged by both methods for their respective reasons. PCA flags them, because they make the frequency of most ROs change in a similar way from one device to another as direction and slope of the speed gradient changes between devices, cf. Sec. 4.2.1. SPACA flags them, because they create regions on the die where ROs tend to be faster and regions where ROs tend to be slower than the other ROs on the device. That both effects are connected to the speed gradients is supported by the observation that 8 out of the 10 devices with highest Moran's I z-score are also among the 10 devices with highest estimation of second and third principal component, i.e. with strongest speed gradient. Furthermore, all 10 devices with highest Moran's I z-score can be found within the 20 devices with strongest speed gradient.

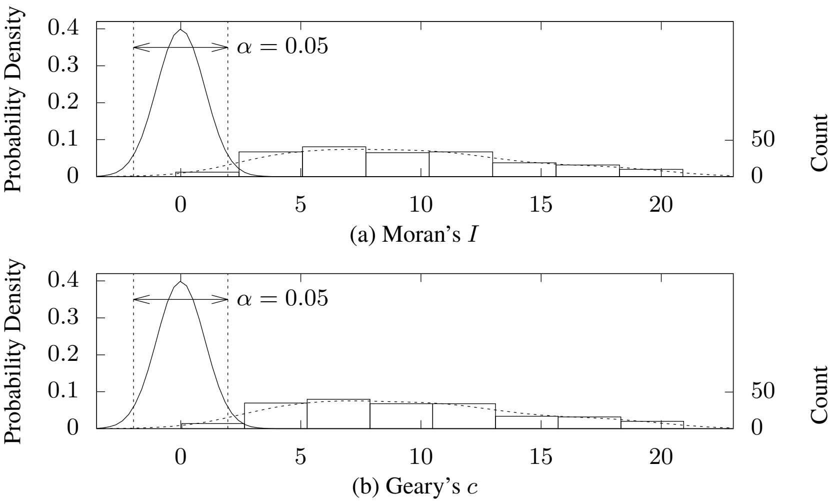
While the RO frequencies of the ROmaiti dataset may be spatially autocorrelated, PCA already showed that the detrimental effect on the binary response bits can be reduced by an adequate bit extraction scheme. To analyze response bits, the join count statistic fits better than Moran's I or Geary's c, because it is



(a) Moran's I

(b) Geary's c

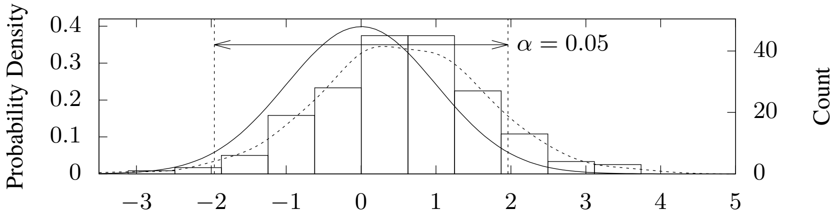
Figure 4.14.: Histograms and KDEs of z-scores under randomization assumption with a rook's move neighborhood for true RO frequencies of ROmaiti dataset.



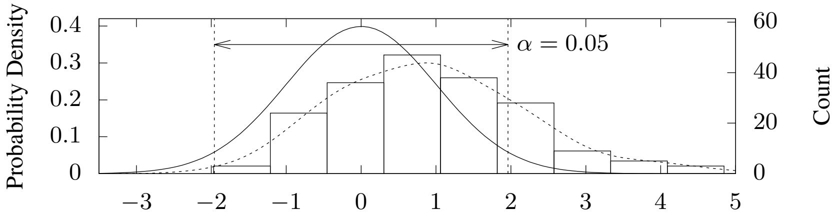
(a) Moran's I

(b) Geary's c

Figure 4.15.: Histograms and KDEs of z-scores under randomization assumption with a rook's move neighborhood for equalized true RO frequencies of ROmaiti dataset.

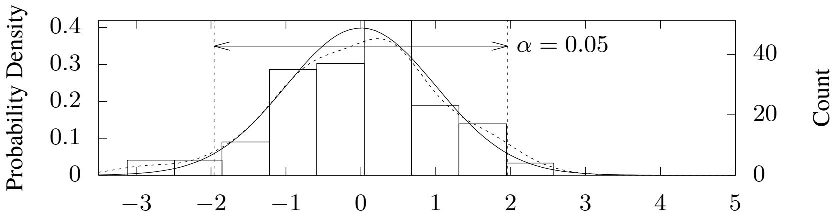


(a) Taxicab limit 1 neighborhood (rook's move)

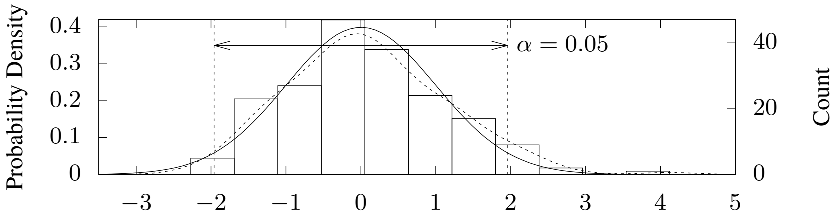


(b) Taxicab limit 3 neighborhood

Figure 4.16.: Join count statistic on response bits from MEPWC on ROmaiti dataset with two different neighborhoods.



(a) Taxicab limit 1 neighborhood (rook's move)



(b) Taxicab limit 3 neighborhood

Figure 4.17.: Join count statistic on response bits from MEPWC on ROmaiti dataset with two different neighborhoods after equalization of expected frequency.

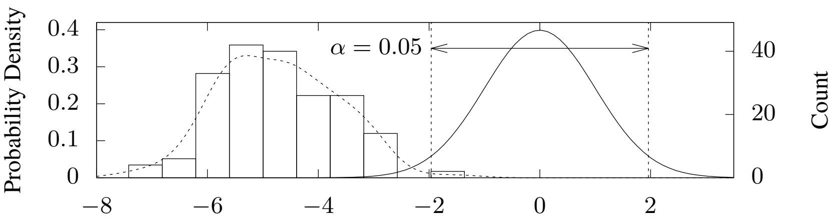


Figure 4.18.: Join count statistic on response bits from overlapping pairwise comparison on ROmaiti dataset with a rook's move neighborhood after equalization of expected frequency.

made for nominal data (such as logical 1s and 0s) instead of interval data (such as frequencies). Fig. 4.16 shows the join count statistic on bits extracted by MEPWC between physically adjacent ROs, cf. (1.13). The spatial autocorrelation nearly disappears – even if the neighborhood is increased from rook's move to a taxicab norm with distance limit 3 for higher sensitivity – because the MEPWC acts as a spatial high-pass filter and effectively reduces the correlation among the response bits. On the one hand, this means that underlying layout issues cannot be detected anymore. On the other hand, this shows that the consequences of suboptimal layouts can be reduced by good bit extraction schemes. With additional equalization of expected frequency between locations, the spatial autocorrelation becomes almost completely invisible, as shown in Fig. 4.17.

However, for pairwise comparison to act as spatial high-pass filter, it needs to be done in a mutually exclusive way. If, for example, response bits are created in an overlapping way such as according to (1.12), negative spatial autocorrelation arises. The reason is that every RO, except for the first and last RO in a row, is used for two response bits, where it has contrary effects because it is on different sides of the comparison operator. Thus if an RO has a frequency above the mean, it increases the probability for the first bit to turn out as 1 and at the same time increases the probability for the second bit to turn out as 0. The result is strong negative spatial autocorrelation even after equalization of expected frequency between locations as shown in Fig. 4.18.

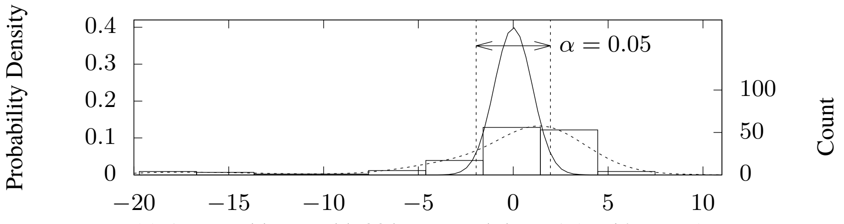
SRAMxmc The SRAMxmc dataset provides an interesting corner case for SPACA, as it contains both positive and negative spatial autocorrelation on different axes. The dataset originates from COTS SRAM on an MCU, so precise layout information is not available as for the ROmaiti dataset analyzed above. However, as the MCU has a native word with 32 bit, common layout techniques suggest a line width of either 32 bit or 64 bit. Note that for this test, a memory line corresponds to a vertical column in the assumed layout. Both assumptions have been tested using the join count statistic with a taxicab neighborhood and a distance limit of one and three to also detect dependencies that do not affect the closest neighbors.

Histograms of the z-scores are shown in Fig. 4.19, where the most noticeable observation is the extremely long tail of z-scores for a line width of 64 bit. Therefore, the SND, which would be unrecognizable due to axes scaling, is omitted. The large z-scores result from the alternating pattern in probability for a 1 that is a known defect of the dataset, cf. Sec. 1.9.4, because with an assumed line width of 64 bit, this make the upper half of the array have an average 60% 1s in it, whereas the lower half contains about 40% of all 1s. A rotated bitmap plot of this behavior can be found in Fig. 3.16. Like the unequal mean frequencies in the ROmaiti dataset, this appears as positive spatial autocorrelation in the SPACA tests.

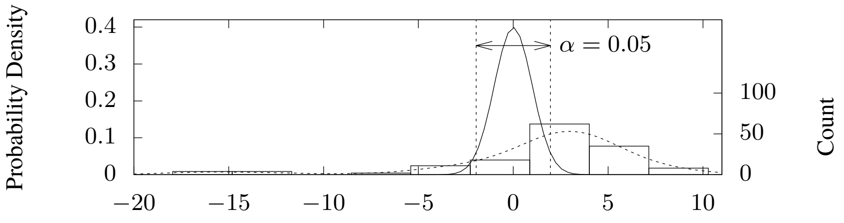
Considering this, the plots for a line width of 32 bit surprisingly show much fewer spatial autocorrelation. The reason for this, as mentioned in the beginning, is that positive and negative spatial autocorrelation cancel each other out in this particular case. A line wrap after every 32 bit leads to an alternating pattern across lines, i.e. negative spatial autocorrelation, whereas within a line all response bits tend to have the same value, i.e. positive spatial autocorrelation. The weights of a taxicab neighborhood are based on distance, not direction, and therefore implicitly add up spatial autocorrelation of any kind in any direction. Although this is a pitfall, it is not covered in standard literature on spatial autocorrelation and thus may be considered a special case. To overcome this problem, SPACA tests can be repeated with weights that credit distance in different axes unequally, such as the irregular taxicab norm exemplified in Fig. 4.13. The price for this second test is just one multiply-add if it is implemented in combination with a standard test, because the only difference lies in the weights of the adjacency matrix. Even if few degenerated distributions might be constructed where negative and positive spatial autocorrelation can still cancel out, it strongly lowers the risk for this to happen in practice.

Fig. 4.20 shows the result of using unequal weights for different directions by the use of an irregular taxicab norm with axes ratio one half and two and a distance limit of three. If steps within a line of the SRAMxmc dataset cost twice as much as steps between lines, the negative spatial autocorrelation between lines dominates as shown in Fig. 4.20a. Likewise, if steps within a line cost only half, the positive spatial autocorrelation within lines dominates as shown in Fig. 4.20b.

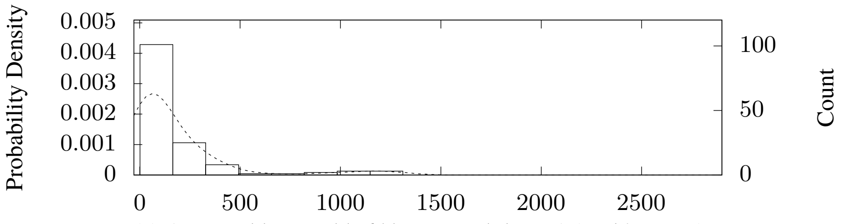
SRAMsu Su et al. [67] analyzed their data for spatial artifacts through a heatmap of Bit-Alias that resembles on-chip layout and a plot of the ratio of 1 responses in each of the eight columns of their layout, which is reproduced in Fig. 4.21 for verification. While the increase of the ratio of 1 responses from left to right for the symmetric design is said to show "gradient effects" [67], they claim on this



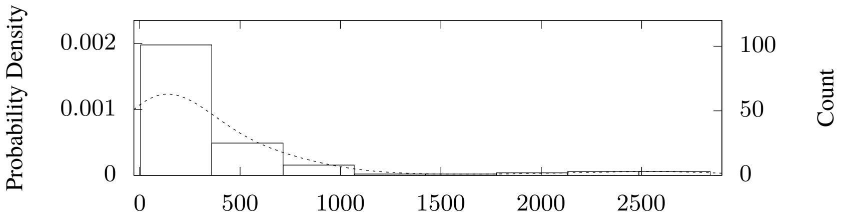
(a) Assumed line width 32 bit, taxicab limit 1 (rook's move)



(b) Assumed line width 32 bit, taxicab limit 3

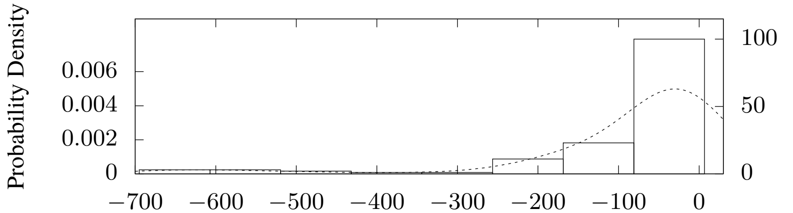


(c) Assumed line width 64 bit, taxicab limit 1 (rook's move)

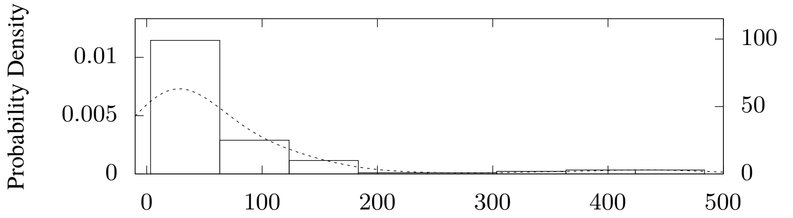


(d) Assumed line width 64 bit, taxicab limit 3

Figure 4.19.: Join count statistic on SRAMxmc dataset for an assumed line width of 32 bit and 64 bit with taxicab neighborhood of limit one and three. Note that the plots for line width 64 bit do not have the SND overlaid as it would be illegible due to extreme axes scaling.



(a) Assumed line width 32 bit, irregular taxicab limit 3 ratio 2



(b) Assumed line width 32 bit, irregular taxicab limit 3 ratio $1/2$

Figure 4.20.: Join count statistic on SRAMxmc dataset for an assumed line width of 32 bit and an irregular taxicab neighborhood. No SND is overlaid as it would be illegible due to extreme axes scaling.

basis that "[n]o noticeable spatial artifacts are present in either circuit" [67]. In addition, they report the conditional probability of the left, right, upper, and lower neighbor to turn out the same respectively the other value in order to find coupling effects. This approach seems similar to SPACA, but does not provide the benefits of a hypothesis test, i.e. one is left with some average probability without guideline how to interpret it or make a decision.

To confirm whether the conclusion of no spatial artifacts holds, the join count statistic with a rook's move neighborhood is applied on the SRAMsu dataset and the result depicted in Fig. 4.22. Indeed, the join count statistic verifies that the PUF candidate circuit developed by Su et al. passes the join count test with all but one device of the common centroid layout, which may be considered a false reject based on $\alpha = 0.05$ and 19 devices per layout being tested. An interesting observation is that while the gradient effect of the symmetric layout would suggest a problem with positive spatial autocorrelation, the majority of devices tends towards negative spatial autocorrelation instead, without this being significant, though. So the slight increase in the number of 1 responses towards the right columns does not cause tests for spatial autocorrelation to trigger, although varying regional bias can do so as shown with the ROmaiti dataset. This suggests that the increase in bias is not significant and provides another example why analysis of regional bias does not replace SPACA.

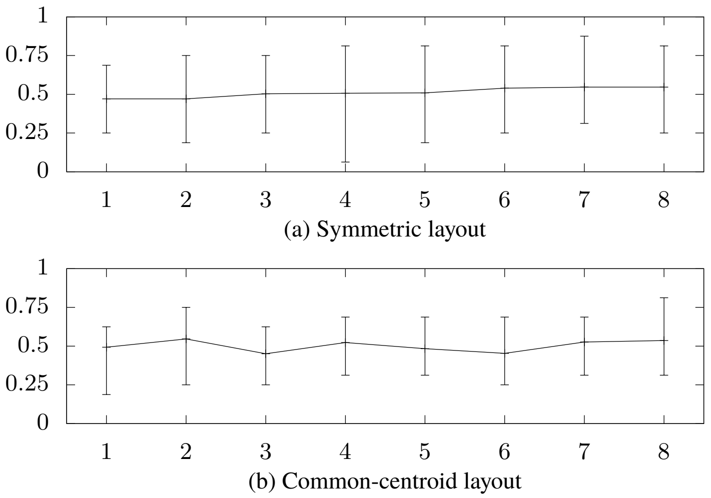


Figure 4.21.: Mean ratio of 1 responses per column and device in SRAMsu dataset. Error bars indicate minimum and maximum ratio among devices.

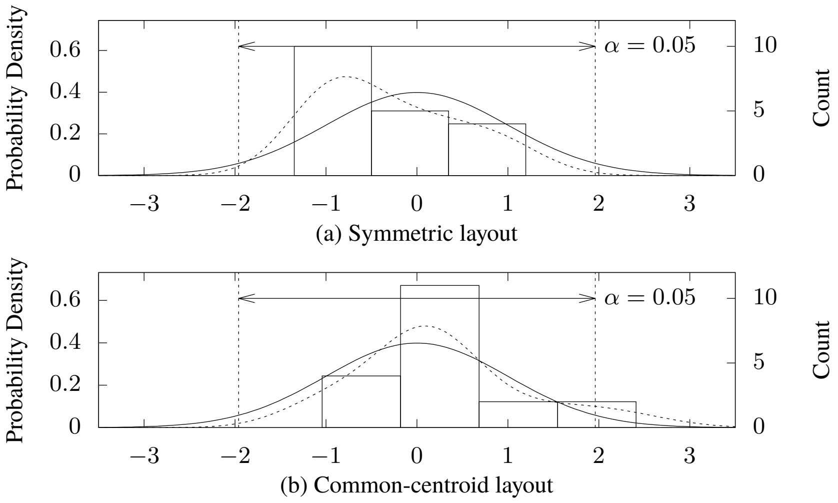


Figure 4.22.: Join count statistic on SRAMsu dataset with a rook's move neighborhood.

Join Count Statistic as Part of a Built-in Self Test

Most of the tests discussed herein are targeted at qualification of a new PUF candidate design at the end of the development phase where full access to the test devices is available. However, that SPACA operates on individual devices and the join count statistic uses rather simple logical and mathematical operations makes it furthermore suitable for a built-in self test (BIST) of the final product. A BIST increases the security of a product in multiple ways. Apart from the ability to detect failure and possibly even attacks in the field, it is a necessity for end-of-line production tests that are supposed to detect defunct devices by e.g. an all-zero response, because the response of non-test devices must not leave them at any time to maintain the security goals of a PUF. A BIST for PUFs has already been proposed by e.g. Hussain et al. [99], who used tests from NIST's TRNG test suite. However, to use SPACA for such a BIST was first proposed in [**60**], and its feasibility on an FPGA was investigated in a Bachelor's thesis under my supervision [108].

A block diagram of the implementation is shown in Fig. 4.23. A finite state machine (FSM) controls the overall procedure of the BIST, which starts once the PUF response is available. It begins with a calculation of the HW of the response, which is necessary to determine the expectation and expected variance of the join count statistic. At this point, a test on the HW is added at virtually no cost to quickly identify devices with suspiciously high or low HW, as this suggests a total failure of the PUF circuit, possibly due to an ongoing attack. To avoid that a device fails this test due to aging, an option for a more strict bound could be added for use during end of line production test. If this test is passed, the FSM initiates the calculation of the join count statistic, where the adjacency matrix, so the information which pairs of response bits to compare, is stored in a block random access memory (RAM). The test statistic is then compared to the limits in another block RAM, which are precalculated based on the desired significance level, the known shape of the PUF cell array, and the adjacency matrix that has been programmed into the first block RAM. Since the limits also depend on the exact HW of the response, which is not known in advance, limits for all HWs that would pass the previous HW test are stored in the block RAM and the BIST selects the appropriate limits internally. In the current implementation aimed for development, the BIST is a separate IP core that is given the PUF response and provides status information on whether and why one of the tests failed. In a productive setting, the BIST would rather be implemented in a way that insulates the PUF circuit from the rest of the system and withholds the PUF response in case any of the tests fails to avoid providing any useful information to an attacker.

Since the BIST only has to perform a pass/fail decision, it would create unnecessary computational effort on the FPGA to calculate Z-scores or p-values from the test statistic. Instead, limits for the test statistic itself can be calculated in reverse

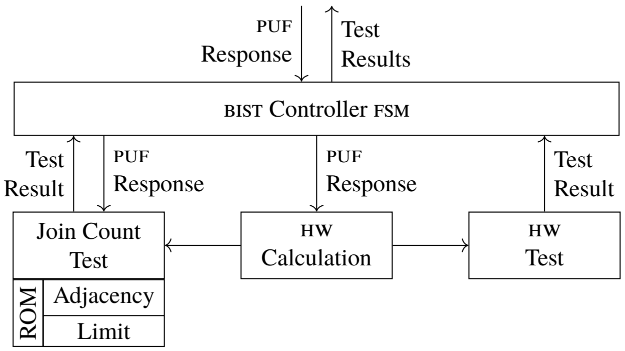


Figure 4.23.: Block diagram of a built-in self test for PUFs that includes a HW test and a join count test on spatial autocorrelation.

based on the desired significance level α of the test via

$$\left|\underset{\mathrm{NF}}{\mathrm{E}}\left[\overset{\bullet\circ}{J}_d\right] - \overset{\bullet\circ}{J}_d\right| \le \sqrt{2\underset{\mathrm{NF}}{\mathrm{Var}}\left[\overset{\bullet\circ}{J}_d\right]}\,\mathrm{erfc}^{-1}(\alpha) \tag{4.50}$$

and stored as a look-up table in a block RAM as described above. The size of the look-up table can be kept low by utilizing two factors: First, by the symmetry of expectation and expected variance with regard to the HW of the response, i.e. their value for m_d is identical to that for $1 - m_d$ and thus the look-up table only needs to contain entries for 0 to $\lceil B/2 \rceil$. Second, by omission of limits for HWs that would already fail at the first test for suspiciously low or high HW. Since the entry corresponding to the device's particular HW is selected by the BIST internally, the same look-up table can be programmed into every device. Another benefit of this approach is that integer arithmetic for the FPGA implementation is sufficient if the adjacency matrix contains only integer weights as with e.g. rook's move.

The prototype of the BIST has been implemented on a Xilinx XC7A35T. A UART connection to a computer provides control signals, responses to test, and a return channel for the test result. This way, 20 000 uniformly sampled artificial PUF responses have been run on the prototype and the results successfully been compared to those from a reference implementation in MATLAB. Resource utilization on the FPGA as reported by the design tool can be found in Tbl. 4.5. If the HW test failed for a PUF response, the BIST completes in 1.92 µs, otherwise the BIST requires 12.4 µs. While this runtime may induce a noticeable delay for PUFs with fast time to response such as SRAM PUFs, it is negligible for e.g. RO PUFs, since they typically require tens or hundreds of milliseconds to produce a response.

Resource type	Utilization (Modules)				Total
	FSM	HW Calculation	HW Test	Join Count Test	
LUT	8	113	7	413	541
FF	525	35	4	55	619
BRAM	0	0	0	6	6

Table 4.5.: Resource utilization of the built-in self test for PUFs on a Xilinx XC7A35T. Reproduced from [108].

4.3. Probability Distribution of PUF Responses With Multivariate Models

As a metric of its own and as prerequisite for the following section, it is worthwhile to investigate the PMF of the actual PUF responses. A direct measurement of the probability for each element of the response space would require $D \gg 2^B$ sample devices to obtain a sufficient number of samples for each possible response. It is thus infeasible for any realistic response size. This is also the reason why a univariate categorical model is infeasible, confer Sec. 3.5. Approximation of the PMF through a histogram is difficult both due to the B-dimensional support and the cardinality of the response space. For discrete RVs every outcome typically constitutes a bin, but 2^B bins are infeasible to print in a histogram. For continuous RVs, continuous ranges of outcome are typically summarized in a bin under the assumption that the probability density changes only negligible within the range covered by a bin. To perform this in B dimensions is infeasible, though. The attempt to reduce the dimensionality by a transformation to e.g. integer representation $00010100 \mapsto 20$ ignores the probability distribution among dimensions. To capture consecutive integers in a bin may therefore cause responses of very different probability in the same bin and completely void the explanatory power of a histogram.

To evaluate the unpredictability of a PUF candidate, however, one is more concerned about how many responses exist with a certain probability rather than which probability a specific response has. One may therefore utilize the fact that every response in the response space has a finite probability and represent the probability distribution not by the PMF, but its inverse[2], i.e. a function that reflects how many responses with a given probability exist, which is hereby named RMF. The support of the RMF is the continuous range $(0, 1)$, respectively $(-\infty, 0)$ for log-probability, which allows to construct histograms through bins of consecutive probability that contain explanatory power.

[2]Not to be confused with the inverse distribution, which is the PMF of the reciprocal of the RV.

For the univariate Bernoulli model, the RMF can be calculated by a method from Delvaux et al. [36], as shown in the first following subsection to provide an easy to grasp understanding of what the RMF is. A noteworthy contribution of this work is then to calculate the RMF for the multivariate models introduced in Sec. 3.5. To achieve this, a method from SCA [109] is adapted to the field of PUFs and extended to allow error shaping.

4.3.1. Univariate Bernoulli Model

If all response bits are drawn from the same univariate Bernoulli model with $p \neq 0.5$, it is known that the probability of a response only depends on its HW, cf. [36] and Sec. 2.8.2. So the 2^B PUF responses can be partitioned into $B+1$ groups φ_j, where $j \in \{0, \dots, B\}$. Each group contains $|\varphi_j|$ responses with the same probability $\overset{\varphi}{p}_j$ and $\overset{\varphi}{p}_j > \overset{\varphi}{p}_{j+1}$. From these groups, the RMF may be written as

$$\overset{\varphi}{f}_p(p') = \sum_{j=0}^{B} \frac{|\varphi_j|}{2^B} \delta\left(p' - \overset{\varphi}{p}_j\right) \tag{4.51}$$

with probability p', respectively

$$\overset{\varphi}{f}_p(q') = \sum_{j=0}^{B} \frac{|\varphi_j|}{2^B} \delta\left(q' - \log_2\left(\overset{\varphi}{p}_j\right)\right) \tag{4.52}$$

with log-probability q', where in both cases $\delta(\cdot)$ is the Dirac-delta function. The RMF is therefore a series of Dirac-delta impulses at the probability of occurence of each group and with weight according to the relative amount of responses with that probability. The RMF therefore is a kind of pseudo-PDF, in the sense that it is a generalized non-negative function whose integral over the entire support is unity, but that has a (log-)probability instead of a sample space as support and returns a proportion rather than a probability.

Histograms of the RMF under this model for response size $B = 8$ and selected values of p are given in Fig. 4.24. Except for the pathological case of $p = 0.5$, the plots contain nine Dirac-delta impulses. Since the cardinality of the groups is identical due to B being the same for all plots, the height of the impulses remains the same. Their location varies according to the variation in p. Compared to the case of $p = 0.5$, which reduces to a single Dirac impulse at $\overset{\varphi}{q} = \log_2\left(\overset{\varphi}{p}\right) = -B$ that contains all 2^B responses, a deviation of p causes the majority of responses to become *less* probable. This, however, comes at the price that the remaining minority of responses becomes correspondingly *more* probable, which impairs

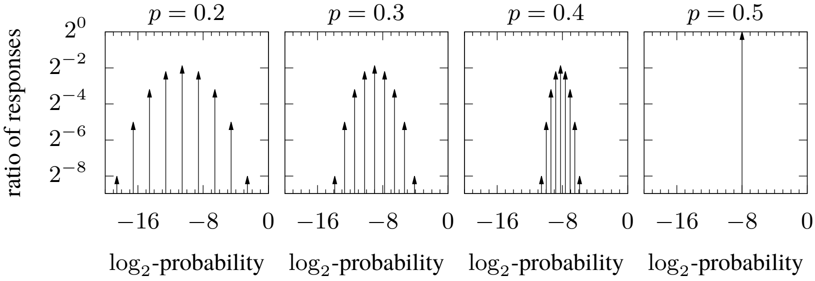


Figure 4.24.: Exemplary histograms of RMF under univariate Bernoulli model for an 8 bit response and selected values of p. Histogram shape for p and $1 - p$ is identical.

the required unpredictability of PUF responses. For example, with the strongest bias depicted in Fig. 4.24, $p = 0.2$, the probability to observe the all zero response $\underline{x} = 00000000$ is $\overset{\varphi}{p}_0 = 0.168$, which means that in a large production lot, approximately a sixth of all devices will feature this response.

4.3.2. Multivariate Bernoulli Model

Under the multivariate Bernoulli model, each response bit position may have a different success probability, thus the probability of a PUF response does not only depend on *how many* response bits turned out in the favorite or unfavorite value, as in the univariate Bernoulli model, but *which*. However, since the random processes are considered independent for each response bit position, the probability of any PUF response may still be calculated as a simple scalar product: The B element row-vector of log-probabilities $\underline{q} = \log_2\left(\underline{p}\right)$ represents how probable it is for a device to produce a 1 at a certain response bit position. The corresponding vector of log-probabilities for a 0 is denoted as $\overset{\neg}{\underline{q}} = \log_2\left(\overset{\neg}{\underline{p}}\right)$. The response candidate is written as $\underline{x}$ and its one-complement is $\overset{\neg}{\underline{x}}$, both are also considered row-vectors. Then the log-probability to observe this response candidate is calculated by

$$q_{\underline{x}} = \underline{q}\,\underline{x}^\top + \overset{\neg}{\underline{q}}\,\overset{\neg}{\underline{x}}^\top , \tag{4.53}$$

which sums up the log-probabilities for a 1 and 0 depending on the response candidate. Although this seems trivial, calculation of all 2^B response probabilities is still infeasible for any practical response size, since $2 \cdot 2^B$ scalar products are required. A solution to this issue can be found in the field of SCA [109], which is adapted to the field of PUFs in this work.

In the field of SCA, a typical attack on e.g. 128 bit AES provides 16 lists of 8 bit subkey candidates together with their (log-)probability of being correct. The *key rank* is the number of incorrect keys an attacker would try before the correct one if she tries every key candidate in decreasing order of probability. To calculate it, the lists for subkeys have to be combined into a single list for complete key candidates. This can be efficiently done according to Glowacz et al. [109] by convolution of histograms. Note that this is different from convolution of the PMFS or PDFS of the RVS in question. This approach, which has been used in Sec. 4.1.1 and performs the convolution in the response space, provides the PMF respectively PDF of the *sum* of these RVS. In this section, the goal is the PMF or PDF of the *concatenation* of the RVS, which is substantially different, e.g. regarding commutativity. The convolution therefore occurs in the domain of log-probability instead.

While the original proof in [109] uses multisets for a mathematically sound explanation, a practical example is given in the following as an easy entrance to the concept. Assume $B = 8$ and some $\underline{p}$ according to Tbl. 4.6 that leads to the RMFS for individual response bit positions on the left hand side of Fig. 4.25. The RMF for the combination of the first two response bit positions is then easily obtained by convolution of the RMFS of the first two responses bit positions in the domain of log-probability. In other words, two copies of the RMF of the second response bit position are placed at positions shifted by the log-probabilities of the RMF of the first response bit position, and each copy is multiplied by one half. This intermediate result is depicted in the upper right part of Fig. 4.25. The shift equals an addition of the log-probabilities, which in turn is equal to the multiplication of probabilities, and this provides the probability of the combined event under the assumption that the individual events are independent. Using the Dirac delta function again, the RMF for individual response bit positions are easily described as

$$f_{p_i}(q') = \frac{1}{2}\left(\delta\left(q' - \log_2(p_i)\right) + \delta\left(q' - \log_2(1 - p_i)\right)\right) \tag{4.54}$$

and due to the properties of the Dirac delta function, their convolution directly provides the required shifted copy operation:

$$f_{\underline{p}}(q') = \mathop{\Huge\ast}_{i=1}^{B} f_{p_i}(q') \tag{4.55}$$

Convolution of RMFS with Dirac impulses is exact, but the computational effort is prohibitive, since the number of Dirac delta functions that are required to describe $f_{\underline{p}}(q')$ grows exponentially while convolving. At the same time, many Dirac impulses in Fig. 4.25 are very close to each other, which suggests to group them together and represent them with a single Dirac delta function of appropriately larger weight. To benefit from the grouping already during convolution, it is performed on the RMFS of the individual response bit positions.

$\underline{p}$	0.235	0.447	0.215	0.689	0.643	0.541	0.348	0.453
$\underline{q}'_1$	−2.091	−1.160	−2.214	−0.538	−0.638	−0.887	−1.521	−1.141
$\underline{q}'_0$	−0.386	−0.856	−0.350	−1.684	−1.484	−1.123	−0.618	−0.871

Table 4.6.: Probabilities and log-probabilities of the example shown in Fig. 4.25. $\underline{p} \leftarrow P \sim \mathfrak{U}(0.2, 0.8)^{\otimes B}$, i.e. each p_b is an independent draw from a uniform distribution between 0.2 and 0.8.

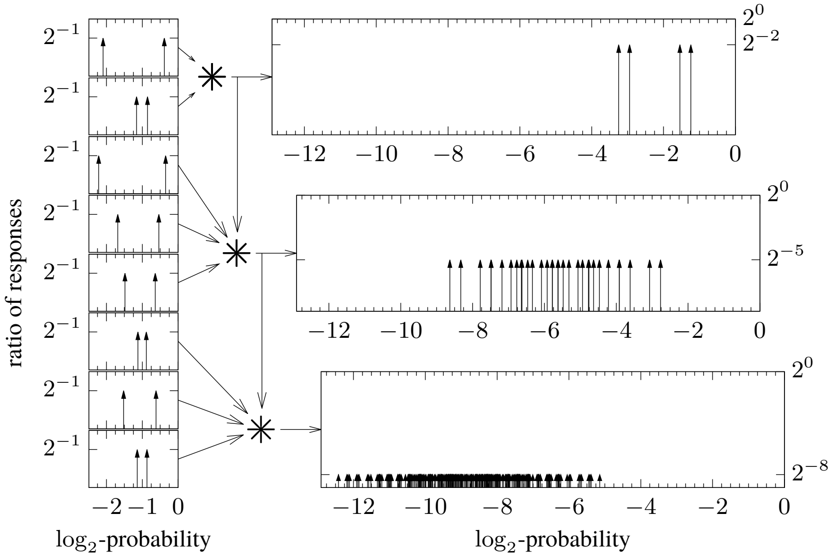


Figure 4.25.: RMFs using Dirac delta function for individual response bit positions and their convolution towards an RMF for the entire response.

This immediately leads to the convolution of histograms of the RMF rather than exact representations of the RMF. As in a regular histogram for continuous RVs, impulses are collected into bins of certain width to describe the RMF more efficiently. The key point here is to use the same bin width, denoted as $\overset{\sqcup}{q}$, for all response bit positions, because the Dirac delta functions that correspond to the bin centers are then identically and evenly spaced, and so will be their convolution. This allows to perform the convolution on the bin count vectors alone, which is a discrete convolution where the output size grows linear, not exponential. More precise, the discrete convolution of an n element and an m element vector results in a $n + m - 1$ element vector. Additionally, discrete convolution is an operation for which highly optimized implementations are available in most statistical and mathematical programming languages. The result of this approach for the same $\underline{p}$ as in Fig. 4.25 is shown in Fig. 4.26. All histograms there use a common bin width $\overset{\sqcup}{q} = 0.25$. For the histograms of individual response bit positions, bins are located so *edges* coincide with tics of x axis. For convoluted histograms, location depends on the sum of central bin values, which causes bin *centers* to coincide with tics of x axis if an even number of histograms are convoluted and bin *edges* to coincide if an odd number of histograms are convoluted. To give a first impression on the accuracy of this method, Fig. 4.26 additionally contains histograms obtained from exact log-probabilities calculated by (4.53) as dashed lines.

A histogram of the RMF provides again a partition – in the mathematical sense – of the 2^B element response space. In the univariate model, the partition was given by the groups φ_j. Here, the groups are denoted as ϕ_j, where $|\phi_j|$ is the bin count, i.e. the number of responses in the group, and

$$\overset{\phi}{q}_j = \log_2\left(\overset{\phi}{p}_j\right) \tag{4.56}$$

is the bin central value as their common log-probability of occurrence. For convenience, the groups are defined to be sorted in decreasing order of probability, i.e. $\overset{\phi}{q}_j > \overset{\phi}{q}_{j+1}$. A downside compared to the grouping under the univariate model is that a group no longer relates to a certain HW and there is no easy way to tell which responses are contained in a particular group. The group of a particular response can be found by (4.58) described below, though. The ability to visualize the probability distribution of the *entire* response space of a PUF candidate with *realistic response size* still provides a remarkable achievement and enables e.g. the extension of the expected conditional min-entropy to multivariate models, cf. Sec. 4.4.

Naturally, the question of accuracy arises when approximate values are further processed. However, since the discrete convolution basically just adds log-probabilities, the error grows only linearly with the number of histograms

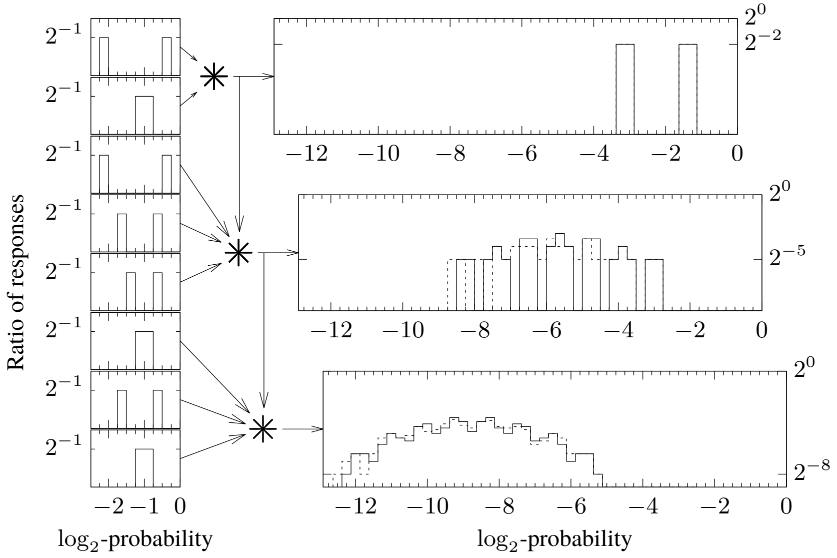


Figure 4.26.: Histograms of RMF for individual response bit positions and their convolution towards a histogram of RMF for the entire response (solid) compared to a histogram of the RMF from exact calculation of log-probabilities (dashed).

convoluted. This means that for each possible response $\underline{x}$, the error between its actual log-probability and its estimated log-probability based on the central values of the corresponding bins is at most

$$\overset{\bar{\wedge}}{q} = \pm k \frac{\overset{\sqcup}{q}}{2}, \tag{4.57}$$

where k is the number of convoluted histograms and $\overset{\sqcup}{q}$ is the bin width. Since bins are represented by their central value, the actual probability of a response within a bin is at most half the bin width away from the bin central value. This error adds up linearly with the number of convoluted histograms, i.e. with the number of RVs or response bit positions for the multivariate Bernoulli case. For the original application of key rank estimation among SCAs, the error limits are $\pm k\overset{\sqcup}{q}$, though. The reason is that the error affects every key candidate, not only the correct one, thus incorrect keys may slide into more probable bins and the correct key into less probable bins, which leads to a triangle inequality, see [109] for details.

For a more precise investigation of the error made by histogram convolution, (4.53) can be adapted to use bin central values as probabilities. With element-wise

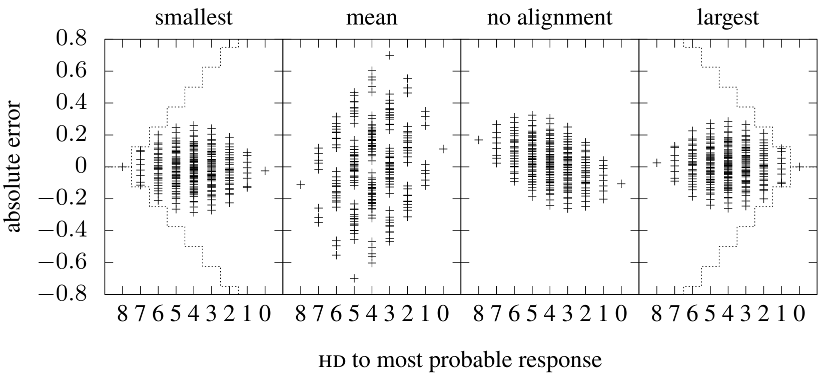


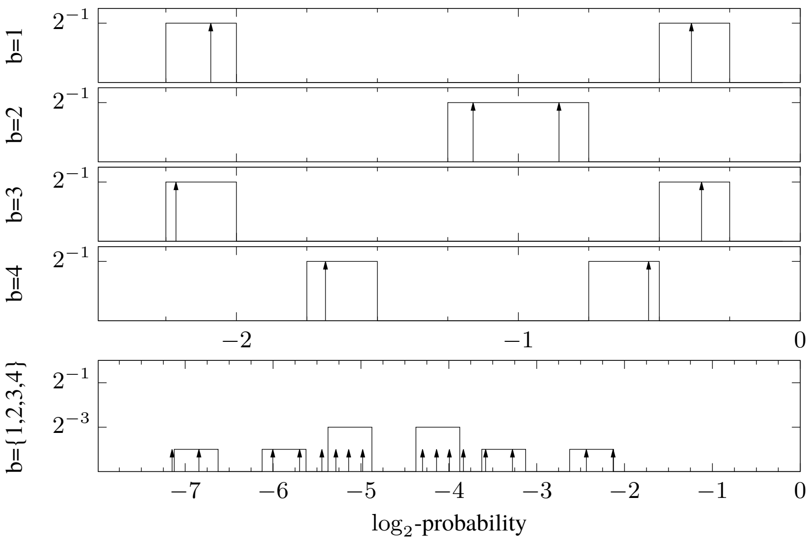
Figure 4.27.: Difference in estimated log-probability of a response between convolution of histograms (4.58) and exact calculation (4.53) for three types of bin alignment, sorted by HD to the most probable response. Dashed lines indicate the improved error bound by alignment of bins.

substitution to bin central values denoted as $[\cdot]$, the estimated log-probability by convolution for a given response $\underline{x}$ is

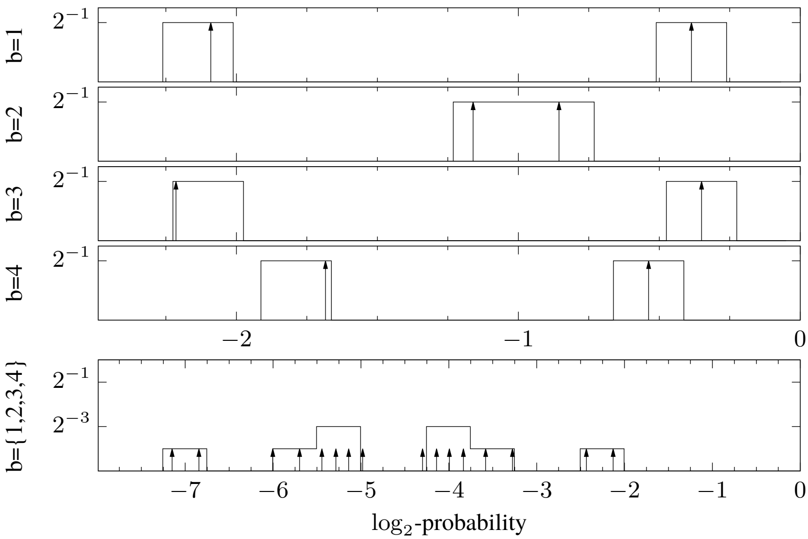
$$[q_{\underline{x}}] = [\underline{q}]\underline{x}^\top + [\overset{\neg}{\underline{q}}]\overset{\neg}{\underline{x}}^\top . \tag{4.58}$$

The difference of (4.58) to (4.53) for the convolution in Fig. 4.26 is shown in Fig. 4.27 as “no alignment”. Each cross corresponds to a response, which is located on the x axis according to its HD to the most probable response and shows on the y axis the difference in log-probability that occurs due to calculation based on bin central values. Although the actual error happens to be quite small, it can be as large as ± 1 according to (4.57) with $k = 8$, $\overset{\sqcup}{q} = 0.25$ and it is infeasible to calculate the exact error for real-world response sizes.

However, since the only requirement to apply this method is to use the same bin *width* for all histograms, not necessarily the same bin *location*, this provides a degree of freedom that can be used to improve the error bound over that from [109]. The idea is to locate the bins of each ingoing histogram individually in a way that the central value of the rightmost bin coincides with the largest log-probability of the corresponding RV respectively response bit position. Therefore no binning error is made for the most probable response of each bit position and their convolution will thus also be free of binning error, i.e. be centered within the corresponding bin. In other words: With this alignment, the most probable response chooses exactly those elements from $[\underline{q}]$ and $[\overset{\neg}{\underline{q}}]$ in (4.58), where the



(a) No alignment with data: Dirac impulses are somewhere in their bin



(b) Rightmost alignment: Dirac impulse in rightmost bin is always centered

Figure 4.28.: Options to align bin central value with actual log-probability to reduce binning error. First four response bit positions and their convolution is shown.

bin central value equals the actual log-probability. This approach is shown in Fig. 4.28 for the first four response bit positions of the previously used example. Without alignment, the Dirac impulses that indicate the exact log-probability of the corresponding outcomes for each bit position are located somewhere in their bin, which means binning error occurs if the bin central value is used to approximate their log-probability. This error propagates during convolution, so the Dirac impulse of the most probable response is not centered within the rightmost bin either. Approximation of the log-probability of the most probable response by the bin central value of the rightmost bin thus comes with an error. If the bins are aligned, though, the bin central value of the rightmost bin always matches the actual log-probability, which is equivalent to the Dirac impulse being centered in their bin. Consequently, the rightmost Dirac impulse in the convolution result is also centered within the rightmost bin. Thus the log-probability of the most probable response can be found by the bin central value of the rightmost bin.

To evaluate the unpredictability of PUFs or to estimate the min-entropy of a set of outcomes, alignment with the most probable response is adequate. The error is then minimal for the most probable response and increases towards less probable responses. However, the same approach can be used with alignment on the leftmost bins, which means the log-probability of the least probable response can be estimated exactly and the error increases towards more probabale responses.

To calculate the log-probability from the convoluted histograms, the central values need to be known. The central value of the leftmost bin of the output histogram is simply the sum of the central values of the leftmost bins of the ingoing histograms. The remaining bins of the output histogram follow with the chosen bin width until the rightmost bin, whose central value is the sum of the central values of the rightmost bins in the ingoing histograms.

While the aligned response is guaranteed to be estimated correctly, the amount of error that may occur for other responses depends on how often an unaligned outcome among the ingoing histograms is chosen. So the HD of a response to the aligned response determines how often an element *with* binning error is chosen from $[\underline{q}]$ and $[\overset{\urcorner}{\underline{q}}]$ in (4.58). The effect is demonstrated in Fig. 4.27, where the same log-probabilities as before are binned with the same bin width, but different alignments of the bins. The figure shows the absolute error in log-probability, calculated as (4.58) minus (4.53) for each response, over the response's HD to the most probable response. The error bound (4.57) from [109] would in this case limit the error to ± 1. With alignment of bin central values, the bound can be tightened to

$$\overset{\overline{\wedge}}{q}_{\underline{x}} = \frac{\overset{\sqcup}{q}}{2} \sum_{b=1}^{B} x_b \oplus x'_b, \tag{4.59}$$

where $\underline{x}'$ is the aligned response, i.e. the most probable response if bins are aligned with the largest log-probabilities and the least probable response if bins are aligned with the smallest log-probabilities. Thus for each response bit position where the response chooses the non-aligned bin, the bound increases by half the bin width, since this is the maximum error that may occur. It is visualized in Fig. 4.27 as dotted line. Note that alignment of bin central values with the mean of both log-probabilities – in attempt to equally spread the binning error – deteriorates the accuracy of estimation although one may intuitively assume the opposite. Since the error is not only increased, but also clustered with bin width distance, the probable cause is that this kind of alignment increases the probability that a convolution result ends up in an adjacent rather than the correct bin.

To further reduce the error made by histogram convolution, one may either use a smaller bin width, increasing computational effort, or reduce the number of convoluted histograms. The latter can be achieved by using a hybrid approach, where log-probabilities of subresponses of feasible size are calculated exactly, then binned into histograms and convoluted. This leads to the third statistical model dealt with in this work, the multivariate categorical model.

4.3.3. Multivariate Categorical Model

The convolution of histograms is not limited to Bernoulli distributions, but may be applied to any set of – not necessarily identical – input distributions, as long as they can be represented by a histogram. In theory, this holds for all discrete probability distributions. In practice, it is feasible as long as the vectors of the histograms are of feasible size, i.e. the distance in log-probability between the most probable and the least probable event for any input distribution is small enough. This is automatically the case for realistic sample sizes if outcomes that are never observed are handled separately rather than with a Dirac impulse at negative infinity. The method can thus be readily applied also under the multivariate categorical model with the sole difference that a RV may have more than two outcomes. The histograms to be convoluted are still built for each RV individually from its respective RMF. This case is in fact even closer to the original application from [109], since the lists of probability for subkeys can be considered as empirical categorical distributions.

The improved error bound also remains applicable, though care must be taken to operate on RVs, which no longer are identical to response bit positions. As for the multivariate Bernoulli model, one may align each ingoing histogram to have a bin central value coincide with the largest – or smallest – log-probability. Following the same arguments as above, if bins are aligned to the largest log-probabilities in all ingoing histograms, the estimated probability for the most probable response will be exact again. Likewise, the estimated probability for the least probable response will be exact if bins are aligned with smallest log-probabilities. The error bound for other responses still increases linearly with the number of input histograms where an unaligned element was chosen, though a histogram corresponds to more than one response bit position now. As example, consider a 512 bit PUF response $\underline{x}$ that is modeled as $\Lambda = 32$ subresponses $\underline{z}_\lambda$ of 8 bit, each with its own 256 outcome categorical distribution. There exist $255\binom{32}{1}$ possible responses that differ in only a single subresponse $\underline{z}_\lambda$ from e.g. the most probable response. Since each categorical distribution is independently binned with bin width $\overset{\sqcup}{q}$ and under the assumption that bins are aligned with the largest log-probability, the estimated probability by convolution, cf. (4.58), for these responses contains only one element with non-zero binning error, and this error is at most half a bin width. Consequentially, the error for the $255^2\binom{32}{2}$ responses that differ in two subresponses, the error is at most twice the former. Further extension is trivial. Thus, the improved error bound (4.59) applies with the sole change that the HD has to be calculated based on the number of unequal subresponses rather than unequal bit positions:

$$\overset{\bar{\wedge}}{q}_{\underline{x}} = \frac{\overset{\sqcup}{q}}{2} \sum_{\lambda=1}^{\Lambda} \begin{cases} 1 & \underline{z}_\lambda \neq \underline{z}'_\lambda \\ 0 & \text{otherwise} \end{cases} \tag{4.60}$$

In comparison to (4.59), the HD operates here on subresponses $\underline{z}_\lambda$ rather than response bit positions.

4.3.4. Examples

To demonstrate the capabilities of the histogram convolution approach, the RMFs of the real-world datasets introduced in Sec. 1.9 are calculated. A multivariate Bernoulli model with one RV per response or identifier bit position was used, where the success probability was estimated from MoD respectively MoK. Convolution was performed with a bin width of 0.01 to minimize the quantization error. Since the resulting RMFs have too many bins to be plotted, they are then summarized into 80 bins for easier visualization in figs. 4.29 to 4.32. Runtime in a virtual machine on a commodity computer was less than one minute for all RMFs.

All RMFs are unimodal with their peak slightly left of the log-probability of an ideal PUF with the corresponding response size. The latter is indicated as a dashed line in the plots. Note that the further the mode is shifted to the left, the more predictable the PUF candidate is, because the sum of probabilities among all responses is always unity. If the mode is further to the left, this means the majority of responses has lower probability, thus there exists a minority of correspondingly more probable responses. The location of the most probable response is indicated by an arrow.

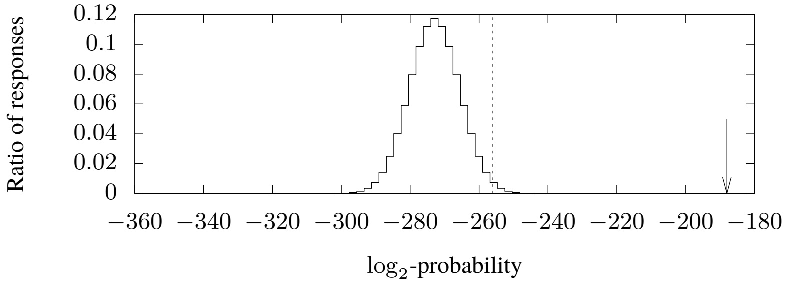


Figure 4.29.: RMF of ROmaiti dataset with bits from MEPWC from convolution under multivariate Bernoulli model with one RV per response bit position.

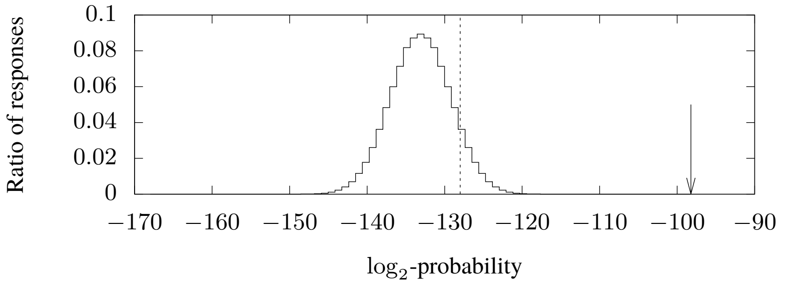


Figure 4.30.: RMF of SRAMsu dataset with common centroid layout from convolution under multivariate Bernoulli model with one RV per response bit position.

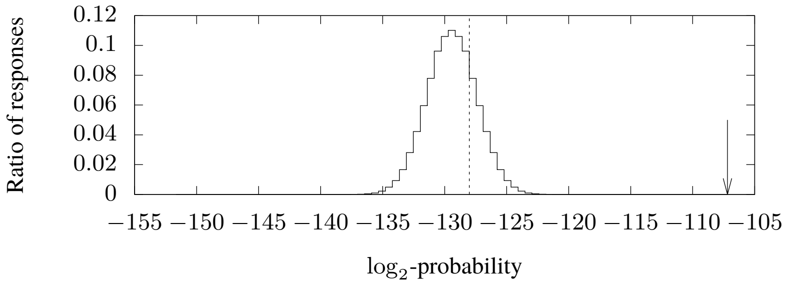


Figure 4.31.: RMF of first device in Ahori dataset from convolution under multivariate Bernoulli model with one RV per identifier bit position.

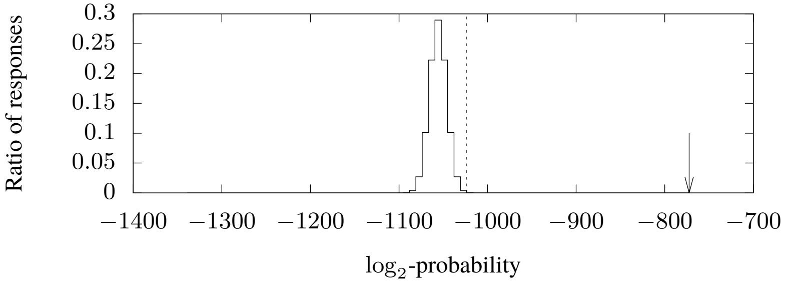


Figure 4.32.: RMF of first 1024 response bit positions of SRAMxmc15 dataset from convolution under multivariate Bernoulli model with one RV per response bit position.

4.4. Expected Conditional Min-Entropy for Multivariate Models

The expected conditional min-entropy $\overline{\mathrm{H}}_\infty(X|Y)$ quantifies the expected min-entropy that remains about a PUF response $\underline{x} \leftarrow X$ in consideration of the information an attacker can gain from the corresponding helper data $\underline{y} \leftarrow Y$. As described in Sec. 2.8.2, exact calculation of $\overline{\mathrm{H}}_\infty(X|Y)$ is only feasible for short ECCs, since it requires at least 2^n operations, where n is the length of a codeword of the ECC. The lower bound on $\overline{\mathrm{H}}_\infty(X|Y)$ developed by Delvaux et al., cf. [36] and Sec. 2.8.2, reduces the computational effort by choosing the $^{2^n}/_{|\mathcal{U}|}$ most probable response words independent of the particular error correction behavior of the ECC and under the assumption that the message $\underline{u} \in \mathcal{U}$ is chosen uniformly at random. Since $^{2^n}/_{|\mathcal{U}|}$ remains infeasible for practically relevant ECCs, Delvaux et al. further required an IID assumption – which is equivalent to this work's univariate Bernoulli model – or an assumption of correlation without bias, to cut the computational effort down to the number of correctable bit errors t. With the results from Sec. 4.3, however, this now becomes feasible for the multivariate Bernoulli and multivariate categorical model and may easily be extended to arbitrary statistical models.

The finding by Delvaux et al. that a lower bound on the expected conditional min-entropy can be based on the probability that any of the $^{2^n}/_{|\mathcal{U}|}$ most probable PUF responses occurs may be written as

$$\overline{\mathrm{H}}_\infty(X|Y) \geq -\log_2\left(\sum_{i=1}^{\frac{2^n}{|\mathcal{U}|}} \mathrm{P}\left(X = \underline{x}_i\right)\right), \tag{4.61}$$

where $\underline{x}_i$ denotes the i^{th} most probable PUF response. With an IID assumption or the univariate Bernoulli model, the PUF responses can be partitioned into groups φ_j, cf. Sec. 4.3, which allows the simplification

$$\overline{\mathrm{H}}_\infty(X|Y) \geq -\log_2\left(\sum_{j=0}^{t'} \min\left(|\varphi_j|, \frac{2^n}{|\mathcal{U}|} - \sum_{j'=0}^{j-1}|\varphi_{j'}|\right) \overset{\varphi}{p}_j\right), \qquad (4.62)$$

where t' is chosen so that ${}^{2^n}\!/\!{}_{|\mathcal{U}|} \leq \sum_{j=0}^{t'}|\varphi_j|$ to minimize the number of iterations. So instead of a sum on individual responses, the t' most probable groups are iterated, where the probability that a group member occurs is weighted with the group's cardinality respectively the remaining amount of responses. To transfer this approach to multivariate models, it is sufficient to replace the groups φ_j with the groups ϕ_j as defined on Page 206. While groups φ_j are defined by the HD of the contained responses to the most probable response, groups ϕ_j relate to bins in the RMF histogram that describes the probability distribution of responses, which is the result of a convolution of RMF histograms of subresponses.

The number of groups to incorporate in the sum depends on their cardinality $|\varphi_j|$ respectively $|\phi_j|$. $|\varphi_j|$ follows a simple binomial coefficient, but $|\phi_j|$ depends on the distribution and the chosen bin width of the pseudo-histograms, which makes it less predictable. The ability to influence cardinality by bin width allows to choose a bin width that results in a feasible number of groups, though.

The representative probability $\overset{\phi}{p}_j$ of a group ϕ_j is the bin central value, which means that in contrast to $\overset{\varphi}{p}_j$ it does not exactly match the probability of each response in the group. Furthermore, a response may end up in a different group by error propagation through convolution. The impact of this can be minimized by alignment of bins with the most probable subresponses as described on Page 208, because it minimizes the error for the most probable responses that contribute to the expected conditional min-entropy. Still, the use of $\overset{\phi}{p}_j$ turns the bound into a mere approximation. To maintain a strict bound, the maximum error has to be added to a group's representative probability. The maximum error in log-probability $\overline{\hat{q}}$ follows from (4.57) or (4.59). So $\overset{\varphi}{p}_j$ would be replaced by $\left(\overset{\phi}{p}_j + 2^{\overline{\hat{q}}}\right)$ in (4.62).

4.4.1. Examples

To illustrate the advantage of the RMF histogram convolution approach, the expected conditional min-entropy and some of its bounds are tabulated in the following for multiple datasets and types of ECC. The type of ECC is denoted by the common (n, k, t) triple for BCH codes, where n is the codeword length, k the message length, and t the number of bit errors guaranteed to be corrected. Repetition codes are denoted by their codeword length (n), since the number of message bits is one by definition and the number of correctable bit flips is $\lfloor (n-1)/2 \rfloor$.

Each of the following tables corresponds to a dataset and for each dataset, the selection of ECC codes is adapted according to the size of the PUF response. As the size of the PUF response is not always a multiple of the codeword size, the tables list the number of PUF response bits consumed, the number of ECC blocks involved, and the number of key bits that may be stored if that combination would be used in a key storage application. For the ROmaiti, SRAMsu, and Ahori datasets, all values in a table have been calculated in less than 10 s in a virtual machine on a commodity computer, and for the SRAMxmc dataset in about 30 s, which proves the superior efficiency of the histogram convolution method over previous approaches such as the grouping bound in [**85**].

The univariate bound uses the univariate Bernoulli model and (4.62) with groups φ_j. The multivariate results are based on the MoD for the single-challenge PUF datasets respectively the MoK for the Ahori dataset. The convolution approximation uses (4.62) with groups ϕ_j and $\overset{\phi}{p}_j$ from RMFs of individual response bit positions with log-probability bin width 0.01. The convolution bound uses the same as before, but with $\left(\overset{\phi}{p}_j + 2^{\bar{\hat{q}}}\right)$. The columns of the following tables denoted as product bound also use (4.62), but with response probabilities from exact calculation based on MoD respectively MoK, i.e. without binning of RMFs. The product exact value finally uses exactly calculated response probabilities of the exact set of error vectors of the corresponding ECC with (2.69). Both product based methods involve 2^n operations for each block of ECC and are thus feasible with today's desktop computers only for small ECCs with codeword size less than 26 bit. In all calculations, the independence of the individual ECC blocks has been considered, i.e. if some response bit positions are so heavily biased that a negative entropy would be estimated for the block they are in, the entropy of the block is considered as zero.

As the examples show, the multivariate convolution approximation is in virtually every case the tightest lower bound on the exactly calculated expected conditional min-entropy, and otherwise overestimates it by at most 0.1 bit according to the tables. Although not a strict bound, it therefore provides more accurate results than the strict bound by addition of the maximum propagated binning error, which is quite conservative.

To use a univariate Bernoulli model though not applicable can lead to an overestimation of expected conditional entropy, e.g. for the ROmaiti dataset of up to 53 bit, see Tbl. 4.7, for the SRAMsu dataset of up to 29.1 bit, see Tbl. 4.8, and for the SRAMxmc dataset of up to 256 bit, see Tbl. 4.10. For the Ahori dataset, the difference between the univariate bound and the multivariate convolution bound is at most 0.9 bit, which suggests that the Ahori dataset fits better to a univariate Bernoulli model than the ROmaiti dataset.

The ROmaiti dataset reaches zero remaining entropy under the multivariate convolution bound for $(127, 8, 31)$ and $(255, 9, 63)$ BCH codes. The SRAMxmc dataset does so for $(255, 9, 63)$, $(511, 10, 127)$, and $(1023, 11, 255)$ BCH codes. For SRAMsu and Ahori, the entire response can be consumed in a single block of BCH code with highest error correction capability and the system still maintains some entropy.

	Response	ECC	Key	Univariate	Multivariate convolution		Multivariate product	
	bits used	blocks	bits	bound	approximation	bound	bound	exact
(3)	255	85	85	77.1	47.1	45.8	47.2	47.2
(7,4,1)	252	36	144	132	88.0	86.7	88.1	88.5
(15,11,1)	255	17	187	173	123	122	123	124
(31,26,1)	248	8	208	193	143	142	–	–
(63,57,1)	252	4	228	213	162	161	–	–
(127,120,1)	254	2	240	224	173	171	–	–
(255,247,1)	255	1	247	231	179	178	–	–
(5)	255	51	51	45.1	22.4	21.2	22.5	22.5
(7)	252	36	36	31.2	12.3	11.1	12.5	12.5
(15,5,3)	255	17	85	76.2	39.8	38.5	39.9	40.7
(31,6,7)	248	8	48	41.2	15.2	13.9	–	–
(21)	252	12	12	9.39	1.44	0.32	1.57	1.57
(63,7,15)	252	4	28	22.8	4.78	3.52	–	–
(127,8,31)	254	2	16	12.1	0.79	0.00	–	–
(255,9,63)	255	1	9	6.09	0.00	0.00	–	–

Table 4.7.: Expected conditional min-entropy for ROmaiti dataset with response bits by MEPWC. Multivariate results are based on the multivariate Bernoulli model with one RV per response bit position and success probability estimated by MoD.

	Response	ECC	Key	Univariate	Multivariate convolution		Multivariate product	
	bits used	blocks	bits	bound	approximation	bound	bound	exact
(3)	126	42	42	41.7	26.1	25.5	26.1	26.1
(7,4,1)	126	18	72	71.6	48.2	47.5	48.2	48.3
(15,11,1)	120	8	88	87.5	61.9	61.3	62.0	62.0
(31,26,1)	124	4	104	103	76.0	75.4	–	–
(63,57,1)	126	2	114	113	85.2	84.6	–	–
(127,120,1)	127	1	120	119	90.5	89.9	–	–
(5)	125	25	25	24.8	12.9	12.2	12.9	12.9
(7)	126	18	18	17.8	8.19	7.56	8.20	8.20
(15,5,3)	120	8	40	39.7	21.9	21.3	22.0	22.3
(31,6,7)	124	4	24	23.7	10.0	9.40	–	–
(21)	126	6	6	5.90	1.36	0.73	1.35	1.35
(63,7,15)	126	2	14	13.8	3.90	3.27	–	–
(127,8,31)	127	1	8	7.84	1.11	0.48	–	–

Table 4.8.: Expected conditional min-entropy for SRAMsu dataset with common-centroid layout. Multivariate results are based on the multivariate Bernoulli model with one RV per response bit position and success probability estimated by MoD.

	Response	ECC	Key	Univariate	Multivariate convolution		Multivariate product	
	bits used	blocks	bits	bound	approximation	bound	bound	exact
(3)	126	42	42	32.0	32.1	31.4	32.0	32.0
(7,4,1)	126	18	72	56.8	56.8	56.2	56.8	56.8
(15,11,1)	120	8	88	71.0	71.1	70.5	71.1	71.2
(31,26,1)	124	4	104	85.1	85.2	84.6	–	–
(63,57,1)	126	2	114	94.2	94.2	93.5	–	–
(127,120,1)	127	1	120	99.6	99.6	99.0	–	–
(5)	125	25	25	17.8	17.8	17.2	17.8	17.8
(7)	126	18	18	12.0	12.0	11.4	12.0	12.0
(15,5,3)	120	8	40	29.4	29.1	28.5	29.1	29.3
(31,6,7)	124	4	24	15.5	15.4	14.8	–	–
(21)	126	6	6	2.95	2.93	2.30	2.91	2.91
(63,7,15)	126	2	14	7.69	7.55	6.92	–	–
(127,8,31)	127	1	8	3.41	3.33	2.70	–	–

Table 4.9.: Expected conditional min-entropy for first device in Ahori dataset. Multivariate results are based on the multivariate Bernoulli model with one RV per identifier bit position and success probability estimated by MoK.

	Response	ECC	Key	Univariate	Multivariate convolution		Multivariate product	
	bits used	blocks	bits	bound	approximation	bound	bound	exact
(3)	1 023	341	341	341	217	212	218	218
(7,4,1)	1 022	146	584	584	394	389	394	396
(15,11,1)	1 020	68	748	748	525	520	525	527
(31,26,1)	1 023	33	858	858	619	614	–	–
(63,57,1)	1 008	16	912	912	669	664	–	–
(127,120,1)	1 016	8	960	960	712	707	–	–
(255,247,1)	1 020	4	988	988	738	733	–	–
(511,502,1)	1 022	2	1 004	1 004	753	748	–	–
(1023,1013,1)	1 023	1	1 013	1 013	762	757	–	–
(5)	1 020	204	204	204	113	108	114	114
(7)	1 022	146	146	146	72.4	67.3	72.6	72.6
(15,5,3)	1 020	68	340	340	195	190	195	198
(31,6,7)	1 023	33	198	198	89.9	84.8	–	–
(21)	1 008	48	48	47.9	12.9	7.86	13.0	13.0
(63,7,15)	1 008	16	112	112	35.8	30.7	–	–
(127,8,31)	1 016	8	64	63.9	11.5	6.44	–	–
(255,9,63)	1 020	4	36	35.9	2.27	0.00	–	–
(511,10,127)	1 022	2	20	19.9	0.03	0.00	–	–
(1023,11,255)	1 023	1	11	11.0	0.00	0.00	–	–

Table 4.10.: Expected conditional min-entropy for first 1024 response bit positions of SRAMxmc15 dataset. Multivariate results are based on the multivariate Bernoulli model with one RV per response bit position and success probability estimated by MoD.

5. Conclusion

Through a detailed overview on existing metrics, it has been shown that metrics for the reliability and unpredictability of PUF candidates are a highly scattered field of research. The large number of metrics and their variants make results hard to compare, because even if the name of the metric is similar or identical, the formula might be different. For example, three different versions of Uniqueness have been discussed in this work. However, the work also shows that groups of metrics can be found where all members can be algebraically transformed into each other, which means they carry the same information about the PUF candidate despite different numeric results. Furthermore, many existing metrics are found to be a series of mean operations, sometimes with an XOR operation in between, which induces the risk of flaws to cancel each other out in the result of the metric.

Among the remaining more complex test methods, two issues have been identified: First, the current standard approach for entropy estimation by compression with the CTW algorithm is surpassed by modern context-mixing based algorithms, such as CMIX and PAQ, which therefore provide closer bounds on entropy. Second, test suites for RNGs, such as NIST SP800-22 or BSI AIS 31, require special caution when applied to PUFs as they are most concerned with the order of 1s and 0s and therefore the way multidimensional response data is flattened into one-dimensional sequences strongly determines the outcome of the tests, which is undesirable.

A statistical approach for unpredictability evaluation of PUFs allows a more realistic view on the achieved accuracy of the results in many previous publications. For this, the expectation and variance of some of the most common metrics – Uniqueness, Bit-Alias, and bitwise entropy – for an ideal PUF are deduced and confidence intervals for Bit-Alias established. A fundamental change in the way unpredictability of PUF candidates is tested has been proposed by use of statistical hypothesis tests specifically tailored to PUFs, which provide more obvious results without subjective interpretation of numeric results. As a starting point, hypothesis tests from the common metrics Uniformity and Bit-Alias and tests for spatial autocorrelation have been developed and demonstrated on real-world datasets.

Finally, the RMF, which describes the number of responses per probability, is introduced as a primary metric for the unpredictability of PUF candidates. It allows to represent the probability distribution of the entire response of a PUF candidate even for realistic response sizes such as 256 bit, 1 024 bit, or more, which has been infeasible before without an IID assumption on the individual response bits. This also allows to calculate the expected conditional min-entropy under multivariate statistical models. Using the four real-world datasets that serve as examples throughout this work, the min-entropy estimated this way is at most 0.1 bit above the exactly calculated value and thus more realistic than estimations using a univariate model.

The findings of this work can help to compare and assess the results obtained by previous metrics and provide new and improved methods for testing the reliability and unpredictability of PUFs.

List of Figures

List of Tables

Abbreviations

AD	Anderson-Darling 142
AES	advanced encryption standard 114, 206
ASIC	application-specific integrated circuit 23, 28, 44, 52, 66, 84, 138, 168
BIST	built-in self test 201–203, 229, 231
BSC	binary symmetric channel 114, 115
BSI	federal office for information security 151, 225
CD	compact disc 15
CDF	cumulative distribution function 51, 130, 161, 164, 166
CI	confidence interval 5, 13, 68, 71, 77, 80, 88, 97, 113, 114, 116, 117, 127, 129, 132–142, 144, 145, 166, 167, 228
CLB	configurable logic block 53
CLT	central limit theorem 77, 88
CMOS	complementary metal-oxide-semiconductor 32
COTS	commercial of-the-shelf 44, 45, 58, 196
CPU	central-processing unit 29
CRB	challenge-response behavior 5, 22–28, 30, 31, 46–49, 61, 97
CRP	challenge-response pair 18–20, 23–32, 34, 35, 42, 47, 119, 120
CT	computed tomography 25
CTW	context-tree weighting 5, 7, 13, 88, 98, 145–147, 151, 225, 231
DPA	differential power analysis 38
ECC	error correction code 21, 22, 33, 90–95, 102, 216–218
EDM	electrical discharge machining 15
EM	electromagnetic 40, 41
FA	fault attack 24, 32, 47

FHD	fractional Hamming-distance 50, 62, 66, 70, 87, 99–101, 141
FHW	fractional Hamming-weight 50, 111, 169
FPGA	field programmable gate array 5, 8, 29, 38, 41–44, 52, 53, 55, 57, 116, 129, 138, 201, 202
FSM	finite state machine 201–203
GVB	Gilbert-Varshamov bound 70
HD	Hamming-distance 11, 12, 19, 22, 58, 61–75, 77, 80, 81, 83, 84, 88, 97–103, 105, 107, 113, 139, 141, 210, 212, 214, 217, 227
HDS	helper data system 32, 91–93, 95, 102, 227
HW	Hamming-weight 12, 26, 27, 29, 47, 77–79, 81, 93, 94, 106, 113, 158–165, 201–204, 208, 229
IC	integrated circuit 15, 17, 18, 20, 22–27, 29, 32, 34, 36, 44, 168
IDE	integrated development environment 38
IID	independent and identically distributed 5, 8, 17, 93–95, 115, 116, 121, 123, 126, 130, 133, 135, 138, 162, 216, 217, 225
IoT	internet of things 31
JTAG	joint test action group 53
KDE	kernel density estimation 192, 194, 229
LUT	look-up table 27, 31, 35
MCU	microcontroller unit 44, 196
MDS	maximum distance separable 91, 95
MEPWC	mutually exclusive pairwise comparison 55, 145, 154, 163–165, 169, 178, 179, 195, 196, 215, 220, 228, 229, 231
MIT	Massachusetts Institute of Technology 16
ML	machine learning 23, 26, 31, 42, 43, 47, 119, 120
MoA	mean along the axis of accesses 5, 12, 106, 107
MoD	mean along the axis of devices 5, 12, 13, 103, 105, 106, 112, 113, 121, 123, 125–128, 135, 138, 141, 145, 214, 218, 220, 221, 223, 228

MoK	mean along the axis of identifiers 12, 105, 106, 214, 218, 222
MOSFET	metal-oxide-semiconductor field-effect transistor 27, 46
MP-3	MPEG-1 audio layer III 15
NIST	national institute of standards and technology 151, 153, 154, 166, 201, 225, 231
NVM	non-volatile memory 27, 32, 33
OWHF	one-way hash function 19, 28, 29
PCA	principal component analysis 5, 7, 169–180, 193, 228, 229
PDF	probability density function 51, 75, 130, 158, 204, 206
PMF	probability mass function 51, 84, 115, 122–126, 130, 141, 158–160, 162–165, 203, 206, 228
POWF	physical one-way function 16–19
POWHF	physical one-way hash function 19
PRF	pseudo random function 17
PRNG	pseudo-random number generator 29, 51, 57
PUF	physical unclonable function 5, 7, 8, 11, 13, 15–52, 57, 58, 61, 62, 64, 66–85, 87–93, 95, 97, 98, 100, 102, 103, 105–109, 111, 113–121, 123–127, 129, 130, 132, 133, 135, 138, 140, 144–147, 149, 151–155, 157–159, 166, 168–173, 181, 183, 187, 188, 191, 192, 199, 201–205, 207–209, 211–218, 225–229, 231
RAM	random access memory 201, 202
RMF	response mass function 5, 157, 203–209, 213–218, 225, 229
RNG	random number generator 80, 97, 111, 120, 151, 153, 154, 166, 225, 231
RO	ring-oscillator 11, 27, 36, 38–41, 46, 52–55, 57, 61, 89, 90, 100, 116, 129, 130, 168, 169, 172–174, 176, 178, 193, 194, 196, 202, 227–229
ROM	read-only memory 32
RV	random variable 19, 50, 51, 86, 87, 92, 103, 115–120, 122, 123, 125, 126, 130–132, 135, 140, 141, 144, 158, 159, 163, 170–172, 174, 191, 203, 206, 208–210, 213–216, 220–223, 228

SCA	side-channel attack 24, 25, 27, 32, 40, 41, 47, 204–206, 209
SEM	scanning electron microscope 25
SF	survival function 161, 164
SIMPL	simulation possible, but laborious 34
SND	standard normal distribution 192, 193, 197–199
SoC	system-on-chip 29
SPACA	spatial autocorrelation analysis 5, 7, 169, 170, 178, 181, 182, 187, 192, 193, 196, 197, 199, 201, 231
SRAM	static random-access memory 11, 17, 20, 27, 43–46, 49, 52, 58, 59, 61, 63, 75, 81, 116, 147, 168, 196, 202, 227
SSE	secondary symbol estimation 149, 151
SVD	singular value decomposition 171
TRNG	true random number generator 5, 7, 21, 29, 46, 48, 86, 87, 115, 166, 201
UMVU	uniform minimum variance unbiased 116, 117, 130, 131
XOR	exclusive-or 33, 42, 43, 62, 75, 90, 104, 105, 113, 115, 183, 225, 227

Symbols

$\underline{\underline{A}}^\top$	Transpose of matrix $\underline{\underline{A}}$
$\overset{\neg}{p}$	One's complement $1 - p$
$\overset{\neg}{101100}$	One's complement 010011
$[p]$	Quantized value of p
$\hat{p}$	Estimated value of p
$\lvert\mathcal{X}\rvert$	Cardinality of set $\mathcal{X}$
$\lvert t\rvert$	Absolute value of t
$\overset{\langle \mathrm{N} \rangle}{I}$	Value of I standardized according to method N
$t \leftarrow T$	t is a realization or observation of random variable T
$\mathrm{erfc}(\cdot)$	Complementary error function
a	Index in dimension of accesses
A	Number of accesses per device
$\mathfrak{A}$	An arbitrary probability distribution
b	Index in dimension of bit positions
b'	Index for a row
b''	Index for a column
B	Number of bit positions in a response
B'	Number of rows
B''	Number of columns
$\mathfrak{B}(N, p)$	Binomial distribution with N draws and success probability p
$\mathfrak{Beta}(\alpha, \beta)$	Beta distribution with parameters α, β
$\mathfrak{Beta}^{-1}(p, \alpha, \beta)$	Quantile function for probability p of Beta distribution with parameters α, β

c	Index in dimension of challenges
C	Number of challenges per device
$\mathfrak{C}(\underline{p})$	Categorical distribution with probability p_1 for category 1, p_2 for category 2, and so on.
$\grave{c}$	Index of a challenge bit
$\grave{C}$	Number of challenge bits

d	Index in dimension of devices
D	Number of devices in dataset

e	Index in dimension of environmental conditions
E	Number of environmental conditions in dataset
$\mathcal{E}$	Set of coset leaders
$\underline{\epsilon}$	A coset leader
$\mathrm{E}[T]$	Expectation of random variable T

f_X	Probability density function of random variable X
f_Y	Probability mass function of random variable Y
F_X	Cumulative distribution function of random variable X
$\overset{\varphi}{f}_p(q')$	Response mass function in log-probability domain under univariate Bernoulli model with success probability p

g	Metrics based on fractional Hamming-distance (FHD) between challenges or identifiers
$\overset{\updownarrow}{g}$	Distribution of intra-class FHD
$\overset{\leftrightarrow}{g}$	Distribution of inter-class FHD
$\overset{\mathrm{Hc}}{g}$	Correctness according to Hori et al. [72]
$\overset{\mathrm{Hd}}{g}$	Diffuseness according to Hori et al. [72]
$\overset{\mathrm{Hu}}{g}$	Uniqueness for a single device according to Hori et al. [72]
$\overset{\underline{\mathrm{Hu}}}{g}$	Uniqueness for all devices according to Hori et al. [72]

h	Metrics based on fractional Hamming-distance (FHD) between devices
$\overset{\updownarrow}{h}$	Distribution of intra-class FHD
$\overset{\shortparallel}{h}$	Mean intra-class FHD
$\overset{\vdash}{h}$	Median intra-class FHD
$\overset{\vdash\!\sim}{h}$	Standard deviation of intra-class FHD
$\overset{\vdash\!\approx}{h}$	Variance of intra-class FHD
$\overset{\ulcorner}{h}$	Maximum intra-class FHD
$\overset{\leftrightarrow}{h}$	Distribution of inter-class FHD
$\overset{\llcorner}{h}$	Minimum inter-class FHD
$\overset{=}{h}$	Mean inter-class FHD
$\overset{\perp}{h}$	Median inter-class FHD
$\overset{\underline{\wr}}{h}$	Standard deviation of inter-class FHD
$\overset{\underline{\wr\wr}}{h}$	Variance of inter-class FHD
$\overset{\lrcorner}{h}$	Maximum inter-class FHD
$\overset{\mathrm{Mr}}{h}$	Reliability according to Merli et al. [33]
$\overset{\mathrm{Mu}}{h}$	Uniqueness according to Merli et al. [33]

H	Shannon entropy
H_∞	Min-entropy
$\overset{\mathrm{Ha}}{H}$	Intra-class entropy according to Holcomb et al. [78]
$\overset{\mathrm{Hi}}{H}$	Identifying information according to Holcomb et al. [78]
$\overset{\mathrm{Hr}}{H}$	Randomness according to Hori et al. [72]
$\overset{\mathrm{Hs}}{H}$	Steadiness according to Hori et al. [72]
$\overset{\mathrm{Mb}}{H}$	Entropy for an attacker that knows bit position bias according to Maes [82]
$\overset{\mathrm{Mg}}{H}$	Entropy for an attacker that knows global bias according to Maes [82]
$\overset{\mathrm{Mi}}{H}$	Entropy for an ignorant attacker according to Maes [82]
$\overset{\mathrm{Mj}}{H}$	Entropy for an attacker that knows the joint probability distribution according to Maes [82]
$\overset{\mathrm{Mp}}{H}$	Entropy considering a model building attack according to Maes [82]
$\overset{\mathrm{b}}{H}_\infty$	Min-entropy per bit
H_0	Null-hypothesis
H_A	Alternative hypothesis
$\mathrm{H}(T)$	Shannon entropy of random variable T
$\mathrm{H}_\infty(T)$	Min-entropy of random variable T
$\overline{\mathrm{H}}_\infty(X\|Y)$	Expected conditional min-entropy of random variable X given random variable Y
$\mathrm{I}(X, Y)$	Mutual information between random variable X and random variable Y
$\overset{\bullet\bullet}{J}$	Join Count statistic with black/black joins counted
$\overset{\bullet\circ}{J}$	Join Count statistic with black/white joins counted
$\overset{\circ\circ}{J}$	Join Count statistic with white/white joins counted

k	Kurtosis of observations
k	Index in dimension of identifiers
K	Number of identifiers per device
l	Index of bit positions in an identifier
L	Number of bit positions in an identifier
m	Fractional Hamming-weight (FHW)
m	Mean of observations
M	Random variable for Bit Alias
μ	Expectation of a random variable
n	Hamming-weight (HW)
$\mathfrak{N}(\mu, \sigma^2)$	Normal distribution with expectation μ and variance σ^2
$\mathfrak{N}^{-1}(p, \mu, \sigma^2)$	Quantile function for probability p of normal distribution with expectation μ and variance σ^2
N	Number of observations
$\overset{\bullet}{n}$	Number of response bits equal to 1
$\overset{\circ}{n}$	Number of response bits equal to 0
p	A probability
$p_{\underline{x}}$	Probability of response $\underline{x}$
$\tilde{p}$	Probability of a bit flip
$\overset{\mathrm{Lv}}{p}$	Interchip variation τ according to Lee et al. [12]
$\overset{\mathrm{Ln}}{p}$	Noise μ according to Lee et al. [12]
$\overset{\varphi}{p}_i$	Probability of a response in subset φ_i
$\overset{\phi}{p}_i$	Representative probability of responses in subset ϕ_i
$\overset{\overline{\phi}}{p}_i$	Worst case probability of responses in subset ϕ_i
$\overset{\boxplus}{p}$	Probability that the response guessed in a model building attack is correct
$\mathrm{P}[T = t]$	Probability to observe t for random variable T

q	A log-probability
$q_{\underline{x}}$	Log-probability of response $\underline{x}$
$\overset{\varphi}{q}_i$	Log-probability of a response in subset φ_i
$\overset{\sqcup}{q}$	Bin width in a histogram in log-probability domain
$\overset{\bar{\wedge}}{q}$	Maximum error in log-probability due to convolution of histograms
$\overset{\phi}{q}_i$	Representative log-probability of responses in subset ϕ_i
r	A correlation
$\overset{\text{Bc}}{r}$	Correlation metric according to Böhm and Hofer [79]
$\overset{\text{Ku}}{r}$	Correlation metric according to Komurcu and Dündar [81]
s	Standard deviation of observations
s^2	Variance of observations
$\mathfrak{T}(\nu)$	Student's t distribution with ν degrees of freedom
$\mathfrak{T}^{-1}(p, \nu)$	Quantile function for probability p of Student's t distribution with ν degrees of freedom
u	A rairly chosen random bit
$\underline{u}$	A uniformly random bit string
U	Random variable for messages in helper data systems (HDSs), all outcomes are equiprobable
$\mathcal{U}$	Set of possible messages for a HDS
$\mathfrak{U}(a, b)$	Continuous uniform probability distribution on interval $[a, b]$
$\mathfrak{U}(\mathcal{U})$	Discrete uniform probability distribution on set $\mathcal{U}$
$\underline{v}$	A codeword in a HDS
$\mathcal{V}$	Set of possible codewords for a HDS
$\mathrm{Var}[T]$	Variance of random variable T

w	A challenge bit to a PUF
$\underline{w}$	A challenge word to a PUF
$\mathcal{W}$	Set of challenges
x	Observed response bit of a PUF
$\underline{x}$	Observed response word of a PUF
$\bar{x}$	True response bit
$\underline{\bar{x}}$	True response word
$\mathcal{X}$	Set of possible responses
X	Random variable for responses
$\tilde{x}$	Run-time noise, i.e. difference between true response and observed response
$\tilde{X}$	Random variable for run-time noise
$\underline{y}$	Helper data word for a HDS
Y	Random variable for helper data
$\mathcal{Y}$	Set of possible helper data words
$\underline{z}$	Subresponse in a response of a PUF
$\underline{\bar{z}}$	True subresponse in a true response
$\mathcal{Z}$	Set of possible subresponses
Z	Random variable for subresponses
z	Critical value in a statistical hypothesis test
θ	An arbitrary parameter of a probability distribution
λ	Index of random variable in a multivariate model
Λ	Number of random variables in a multivariate model
ξ	Continuous response value of a PUF, e.g. frequency of a ring-oscillator (RO)
Ξ	Random variable for continuous response value
$\bar{\xi}$	True continuous response value

σ	Standard deviation of a random variable
σ^2	Variance of a random variable
ϕ	Group of responses with equal approximated probability of occurence
ω	Adjacency weight for spatial autocorrelation analysis (SPACA)
$\underline{\underline{\omega}}$	Adjacency matrix for SPACA

References

References with bold number indicate that I am one of the authors of the referenced publication.

[1] N. Withers, *What is the most powerful microscope?*, accessed 2019-12-03. available from: `https://www.sciencefocus.com/science/what-is-the-most-powerful-microscope/`.

[2] R. S. Pappu, “Physical one-way functions”, PhD thesis, Massachusetts Institute of Technology, 2001.

[3] K. H. Lofstrom, “The Launch Loop - A low cost earth-to-high-orbit launch system”, in *21st Joint Propulsion Conference*. DOI: 10.2514/6.1985-1368. available from: `https://arc.aiaa.org/doi/abs/10.2514/6.1985-1368`.

[4] Wikipedia contributors, *Keith Lofstrom — Wikipedia, The Free Encyclopedia*, [Online; accessed 28-June-2020], 2019. available from: `https://en.wikipedia.org/w/index.php?title=Keith_Lofstrom&oldid=933037626`.

[5] K. Lofstrom, W. R. Daasch, and D. Taylor, “IC identification circuit using device mismatch”, in *2000 IEEE International Solid-State Circuits Conference. Digest of Technical Papers (Cat. No.00CH37056)*, IEEE, Feb. 2000, pp. 372–373. DOI: 10.1109/ISSCC.2000.839821.

[6] Y. Su, J. Holleman, and B. Otis, “A 1.6pJ/bit 96% Stable Chip-ID Generating Circuit using Process Variations”, in *2007 IEEE International Solid-State Circuits Conference. Digest of Technical Papers*, Feb. 2007, pp. 406–611. DOI: 10.1109/ISSCC.2007.373466.

[7] B. Gassend, D. Clarke, M. Van Dijk, and S. Devadas, “Silicon physical random functions”, in *Proceedings of the 9th ACM conference on Computer and communications security*, ACM, 2002, pp. 148–160.

[8] R. D. Yates and D. J. Goodman, *Probability and stochastic processes, A friendly introduction for electrical and computer engineers*, Third edition. John Wiley & Sons, 2014, ISBN: 9781118324561.

[9] F. Armknecht, R. Maes, A.-R. Sadeghi, F.-X. Standaert, and C. Wachsmann, “A Formal Foundation for the Security Features of Physical Functions”, in *2011 IEEE Symposium on Security and Privacy*, IEEE, 2011, pp. 397–412.

[10] M. Pehl, private communication.

[11] R. Maes, P. Tuyls, and I. Verbauwhede, “A soft decision helper data algorithm for SRAM PUFs”, in *2009 IEEE International Symposium on Information Theory*, Jun. 2009, pp. 2101–2105. DOI: 10.1109/ISIT.2009.5205263.

[12] J. W. Lee, Daihyun Lim, B. Gassend, G. E. Suh, M. van Dijk, and S. Devadas, “A technique to build a secret key in integrated circuits for identification and authentication applications”, in *2004 Symposium on VLSI Circuits. Digest of Technical Papers (IEEE Cat. No.04CH37525)*, Jun. 2004, pp. 176–179. DOI: 10.1109/VLSIC.2004.1346548.

[13] U. Rührmair, F. Sehnke, J. Sölter, G. Dror, S. Devadas, and J. Schmidhuber, “Modeling attacks on physical unclonable functions”, in *Proceedings of the 17th ACM conference on Computer and communications security*, 2010, pp. 237–249.

[14] D. C. Ranasinghe, D. W. Engels, and P. H. Cole, “Security and privacy solutions for low-cost RFID systems”, in *Proceedings of the 2004 Intelligent Sensors, Sensor Networks and Information Processing Conference, 2004.*, Dec. 2004, pp. 337–342. DOI: 10.1109/ISSNIP.2004.1417485.

[15] S. Katzenbeisser, Ü. Kocabaş, V. Rožić, A.-R. Sadeghi, I. Verbauwhede, and C. Wachsmann, “PUFs: Myth, fact or busted? A security evaluation of physically unclonable functions (PUFs) cast in silicon”, in *International Workshop on Cryptographic Hardware and Embedded Systems*, Springer, 2012, pp. 283–301.

[16] P. Tuyls, B. Škorić, S. Stallinga, A. H. M. Akkermans, and W. Ophey, “Information-Theoretic Security Analysis of Physical Uncloneable Functions”, in *International Conference on Financial Cryptography and Data Security*, A. S. Patrick and M. Yung, Eds., ser. LNCS, vol. 3570, Springer, 2005, pp. 141–155.

[17] J. Guajardo, S. S. Kumar, G.-J. Schrijen, and P. Tuyls, “FPGA intrinsic PUFs and their use for IP protection”, in *International workshop on Cryptographic Hardware and Embedded Systems*, Springer, 2007, pp. 63–80.

[18] D. Merli, D. Schuster, F. Stumpf, and G. Sigl, “Semi-invasive EM attack on FPGA RO PUFs and countermeasures”, in *Proceedings of the Workshop on Embedded Systems Security*, 2011, pp. 1–9.

[19] D. Merli, J. Heyszl, B. Heinz, D. Schuster, F. Stumpf, and G. Sigl, “Localized electromagnetic analysis of RO PUFs”, in *2013 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, Jun. 2013, pp. 19–24. DOI: 10.1109/HST.2013.6581559.

[20] L. Tebelmann, M. Pehl, and V. Immler, “Side-Channel Analysis of the TERO PUF”, in *Constructive Side-Channel Analysis and Secure Design*, I. Polian and M. Stöttinger, Eds., Cham: Springer International Publishing, 2019, pp. 43–60, ISBN: 978-3-030-16350-1.

[21] B. Skoric, S. Maubach, T. Kevenaar, and P. Tuyls, “Information-theoretic analysis of coating PUFs”, *IACR Cryptology ePrint Archive*, vol. 2006, no. 101, 2006.

[22] V. Immler, J. Obermaier, M. König, M. Hiller, and G. Sig, “B-TREPID: Batteryless tamper-resistant envelope with a PUF and integrity detection”, in *2018 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, Apr. 2018, pp. 49–56. DOI: 10.1109/HST.2018.8383890.

[23] B. Gassend, D. Lim, D. Clarke, M. Van Dijk, and S. Devadas, “Identification and authentication of integrated circuits”, *Concurrency and Computation: Practice and Experience*, vol. 16, no. 11, pp. 1077–1098, 2004.

[24] U. Rührmair, J. Sölter, F. Sehnke, X. Xu, A. Mahmoud, V. Stoyanova, G. Dror, J. Schmidhuber, W. Burleson, and S. Devadas, “PUF Modeling Attacks on Simulated and Silicon Data”, *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 11, pp. 1876–1891, Nov. 2013, ISSN: 1556-6021. DOI: 10.1109/TIFS.2013.2279798.

[25] F. Ganji, S. Tajik, F. Fäßler, and J.-P. Seifert, “Strong machine learning attack against PUFs with no mathematical model”, in *International Conference on Cryptographic Hardware and Embedded Systems*, Springer, 2016, pp. 391–411.

[26] T. Rahman, D. Forte, J. Fahrny, and M. Tehranipoor, “ARO-PUF: An Aging-resistant Ring Oscillator PUF Design”, in *Proceedings of the Conference on Design, Automation & Test in Europe*, ser. DATE ’14, Dresden, Germany: European Design and Automation Association, 2014, 69:1–69:6, ISBN: 978-3-9815370-2-4. available from: `http://dl.acm.org.eaccess.ub.tum.de/citation.cfm?id=2616606.2616692`.

[27] M. Kalyanaraman and M. Orshansky, “Highly secure strong PUF based on nonlinearity of MOSFET subthreshold operation”, *IACR Cryptology ePrint Archive*, vol. 2012, no. 413, 2012.

[28] C. Helfmeier, C. Boit, D. Nedospasov, and J.-P. Seifert, “Cloning physically unclonable functions”, in *Hardware-Oriented Security and Trust (HOST), 2013 IEEE International Symposium on*, IEEE, 2013, pp. 1–6.

[29] A. Maiti, R. Nagesh, A. Reddy, and P. Schaumont, "Physical Unclonable Function and True Random Number Generator: A Compact and Scalable Implementation", in *Proceedings of the 19th ACM Great Lakes Symposium on VLSI*, ser. GLSVLSI '09, Boston Area, MA, USA: ACM, 2009, pp. 425–428, ISBN: 978-1-60558-522-2. DOI: 10.1145/1531542.1531639. available from: `http://doi.acm.org/10.1145/1531542.1531639`.

[30] A. van Herrewege, V. van der Leest, A. Schaller, S. Katzenbeisser, and I. Verbauwhede, "Secure PRNG seeding on commercial off-the-shelf microcontrollers", in *Proceedings of the 3rd international workshop on Trustworthy embedded devices*, ACM, 2013, pp. 55–64.

[31] D. Lim, J. W. Lee, B. Gassend, G. E. Suh, M. van Dijk, and S. Devadas, "Extracting secret keys from integrated circuits", *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 13, no. 10, pp. 1200–1205, Oct. 2005, ISSN: 1557-9999. DOI: 10.1109/tvlsi.2005.859470.

[32] G. E. Suh and S. Devadas, "Physical unclonable functions for device authentication and secret key generation", in *Proceedings of the 44th annual Design Automation Conference*, ACM, 2007, pp. 9–14.

[33] D. Merli, F. Stumpf, and C. Eckert, "Improving the quality of ring oscillator PUFs on FPGAs", in *Proceedings of the 5th Workshop on Embedded Systems Security*, ACM, 2010, p. 9.

[34] R. Maes, A. Van Herrewege, and I. Verbauwhede, "PUFKY: a fully functional PUF-based cryptographic key generator", in *International Workshop on Cryptographic Hardware and Embedded Systems*, Springer, 2012, pp. 302–319.

[35] M. Yu, R. Sowell, A. Singh, D. M'Raïhi, and S. Devadas, "Performance metrics and empirical results of a PUF cryptographic key generation ASIC", in *2012 IEEE International Symposium on Hardware-Oriented Security and Trust*, Jun. 2012, pp. 108–115. DOI: 10.1109/HST.2012.6224329.

[36] J. Delvaux, D. Gu, I. Verbauwhede, M. Hiller, and M.-D. M. Yu, "Efficient Fuzzy Extraction of PUF-Induced Secrets: Theory and Applications", *IACR Cryptology ePrint Archive*, vol. 2015, no. 854, 2015.

[**37**] M. Pehl, F. Wilde, B. Gammel, and G. Sigl, "Qualitätsevaluierung von Physical Unclonable Functions als Schlüsselspeicher", de, in *14. Deutscher IT-Sicherheitskongress*, Bonn, Deutschland, May 2015.

[38] R. Maes, V. van der Leest, E. van der Sluis, and F. Willems, "Secure key generation from biased PUFs", in *Cryptographic Hardware and Embedded Systems–CHES 2015*, Springer, 2015, pp. 517–534.

[39] L. Zhang, X. Fong, C. Chang, Z. H. Kong, and K. Roy, “Optimizating Emerging Nonvolatile Memories for Dual-Mode Applications: Data Storage and Key Generator”, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 7, pp. 1176–1187, Jul. 2015, ISSN: 0278-0070. DOI: 10.1109/TCAD.2015.2427251.

[40] B. Škorić, “A trivial debiasing scheme for Helper Data Systems”, *IACR Cryptology ePrint Archive*, vol. 2016, no. 241, 2016.

[41] M. Hiller, M. Pehl, G. Kramer, and G. Sigl, “Algebraic Security Analysis of Key Generation with Physical Unclonable Functions”, en, in *PROOFS 2016: Security Proofs for Embedded Systems*, Santa Barbara, CA, USA, Aug. 2016.

[42] M. Hiller, A. G. Önalan, G. Sigl, and M. Bossert, “Online Reliability Testing for PUF Key Derivation”, in *Proceedings of the 6th International Workshop on Trustworthy Embedded Devices*, ser. TrustED ’16, Vienna, Austria: ACM, 2016, pp. 15–22, ISBN: 978-1-4503-4567-5. DOI: 10.1145/2995289.2995293. available from: `http://doi.acm.org.eaccess.ub.tum.de/10.1145/2995289.2995293`.

[43] M. T. Rahman, F. Rahman, D. Forte, and M. Tehranipoor, “An Aging-Resistant RO-PUF for Reliable Key Generation”, *IEEE Transactions on Emerging Topics in Computing*, vol. 4, no. 3, pp. 335–348, Jul. 2016, ISSN: 2168-6750. DOI: 10.1109/TETC.2015.2474741.

[44] M. T. Rahman, A. Hosey, Z. Guo, J. Carroll, D. Forte, and M. Tehranipoor, “Systematic Correlation and Cell Neighborhood Analysis of SRAM PUF for Robust and Unique Key Generation”, *Journal of Hardware and Systems Security*, pp. 1–19, 2017.

[45] T. Speers, *PolarFire™ Non-Volatile FPGA Family Delivers Ground Breaking Value: Best-In-Class Security*, accessed 2020-01-28. available from: `https://www.microsemi.com/blog/2018/04/10/polarfire-non-volatile-fpga-family-delivers-ground-breaking-value-best-in-class-security/`.

[46] T. Lu, R. Kenny, and S. Atsatt, “Secure Device Manager for Intel® Stratix® 10 Devices Provides FPGA and SoC Security”, Intel Corporation, Tech. Rep. WP-01252-1.2, accessed 2020-01-28. available from: `https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/wp/wp-01252-secure-device-manager-for-fpga-soc-security.pdf`.

[47] N. Menhorn, “External Secure Storage Using the PUF”, Xilinx Inc., Tech. Rep. XAPP1333 (v1.0), Jun. 26, 2018, accessed 2020-01-28. available from: `https://www.xilinx.com/support/documentation/application_notes/xapp1333-external-storage-puf.pdf`.

[48] F. Armknecht, R. Maes, A.-R. Sadeghi, B. Sunar, and P. Tuyls, “Memory leakage-resilient encryption based on physically unclonable functions”, in *Towards Hardware-Intrinsic Security*, Springer, 2010, pp. 135–164.

[49] U. Rührmair, “Oblivious transfer based on physical unclonable functions”, in *International Conference on Trust and Trustworthy Computing*, Springer, 2010, pp. 430–440.

[50] M. Majzoobi, M. Rostami, F. Koushanfar, D. Wallach, and S. Devadas, “Slender PUF protocol: A lightweight, robust, and secure authentication by substring matching”, in *2012 IEEE Symposium on Security and Privacy Workshops*, IEEE, 2012, pp. 33–44.

[51] M.-D. Yu, D. M’Rahi, I. Verbauwhede, and S. Devadas, “A noise bifurcation architecture for linear additive physical functions”, in *2014 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, IEEE, 2014, pp. 124–129.

[52] M. Pehl, *Lecture Physical Unclonable Functions*, Slides to Topic 1, 2018.

[53] Y. Dodis, L. Reyzin, and A. Smith, “Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data”, in *Advances in Cryptology - EUROCRYPT 2004*, C. Cachin and J. L. Camenisch, Eds., Springer Berlin Heidelberg, 2004, pp. 523–540, ISBN: 978-3-540-24676-3. DOI: 10.1007/978-3-540-24676-3_31. available from: https://link.springer.com/chapter/10.1007/978-3-540-24676-3_31.

[54] M. Potkonjak and V. Goudar, “Public Physical Unclonable Functions”, *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1142–1156, Aug. 2014, ISSN: 1558-2256. DOI: 10.1109/JPROC.2014.2331553. available from: https://ieeexplore.ieee.org/document/6856138.

[55] N. Beckmann and M. Potkonjak, “Hardware-based public-key cryptography with public physically unclonable functions”, in *International Workshop on Information Hiding*, Springer, 2009, pp. 206–220. available from: https://www.cs.ucla.edu/~miodrag/papers/Beckmann_InfoHiding_2009.pdf.

[56] A. Maiti and P. Schaumont, “Improving the quality of a physical unclonable function using configurable ring oscillators”, in *Field Programmable Logic and Applications, 2009. FPL 2009.*, (Aug. 31, 2009), IEEE, 2009, pp. 703–707.

[57] A. Maiti and P. Schaumont, “Improved ring oscillator PUF: an FPGA-friendly secure primitive”, *Journal of cryptology*, vol. 24, no. 2, pp. 375–397, 2011.

[58] Q. Zhang, Z. Liu, C. Ma, C. Li, and J. Jing, “FROPUF: How to Extract More Entropy from Two Ring Oscillators in FPGA-Based PUFs”, *IACR Cryptology ePrint Archive*, vol. 2015, no. 545, 2015.

[59] C. Costea, F. Bernard, V. Fischer, and R. Fouquet, “Analysis and enhancement of ring oscillators based physical unclonable functions in FPGAs”, in *2010 International Conference on Reconfigurable Computing and FPGAs (ReConFig)*, IEEE, 2010, pp. 262–267.

[**60**] F. Wilde, B. M. Gammel, and M. Pehl, “Spatial Correlation Analysis on Physical Unclonable Functions”, *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 6, pp. 1468–1480, Jun. 2018, ISSN: 1556-6013. DOI: 10.1109/TIFS.2018.2791341.

[61] C. D. Yin and G. Qu, “LISA: Maximizing RO PUF’s secret extraction”, in *2010 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, Jun. 2010, pp. 100–105. DOI: 10.1109/HST.2010.5513105.

[62] G. Hospodar, R. Maes, and I. Verbauwhede, “Machine learning attacks on 65nm Arbiter PUFs: Accurate modeling poses strict bounds on usability”, in *2012 IEEE International Workshop on Information Forensics and Security (WIFS)*, ISSN: 2157-4766, 2157-4774, 2012, pp. 37–42. DOI: 10.1109/WIFS.2012.6412622.

[63] J. Delvaux and I. Verbauwhede, “Fault Injection Modeling Attacks on 65nm Arbiter and RO Sum PUFs via Environmental Changes”, *IACR Cryptology ePrint Archive*, vol. 2013, no. 619, 2013.

[64] M. Majzoobi, F. Koushanfar, and S. Devadas, “FPGA PUF using programmable delay lines”, in *Information Forensics and Security (WIFS), 2010 IEEE International Workshop on*, IEEE, 2010, pp. 1–6.

[65] P. H. Nguyen, D. P. Sahoo, C. Jin, K. Mahmood, U. Rührmair, and M. van Dijk, “The interpose PUF: Secure PUF design against state-of-the-art machine learning attacks”, *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 243–290, 2019.

[66] D. E. Holcomb, W. P. Burleson, K. Fu, *et al.*, “Initial SRAM state as a fingerprint and source of true random numbers for RFID tags”, in *Proceedings of the Conference on RFID Security*, vol. 7, 2007.

[67] Y. Su, J. Holleman, and B. P. Otis, “A digital 1.6 pJ/bit chip identification circuit using process variations”, *IEEE Journal of Solid-State Circuits*, vol. 43, no. 1, pp. 69–77, 2008.

[68] M. Bucci and R. Luzzi, “Identification circuit and method for generating an identification bit using physical unclonable functions”, 8,583,710, US Patent 8,583,710, Nov. 12, 2013.

[69] F. Ganji. (2018). PUFmeter: A Property Testing Tool for Physically Unclonable Functions, available from: `https://trust-hub.org/software`.

[70] A. Maiti, J. Casarona, L. McHale, and P. Schaumont, "A large scale characterization of RO-PUF", in *Hardware-Oriented Security and Trust (HOST), 2010 IEEE International Symposium on*, IEEE, 2010, pp. 94–99.

[**71**] F. Wilde, M. Hiller, and M. Pehl, "Statistic-based security analysis of ring oscillator PUFs", in *Integrated Circuits (ISIC), 2014 14th International Symposium on*, IEEE, 2014, pp. 148–151.

[72] Y. Hori, T. Yoshida, T. Katashita, and A. Satoh, "Quantitative and statistical performance evaluation of arbiter physical unclonable functions on FPGAs", in *2010 International Conference on Reconfigurable Computing and FPGAs*, IEEE, 2010, pp. 298–303.

[73] Y. Hori. (2010). Yohei HORI's Web Site - Profile. accessed 2017-09-27, available from: `https://staff.aist.go.jp/hori.y/en/puf/`.

[**74**] F. Wilde, "Large Scale Characterization of SRAM on Infineon XMC Microcontrollers as PUF", en, in *4th Workshop on Cryptography and Security in Computing Systems (CS2 2017) HIPEAC17*, Stockholm, Sweden, Jan. 2017. DOI: 10.1145/3031836.3031839. available from: `https://dl.acm.org/citation.cfm?id=3031839`.

[75] G. Selimis, M. Konijnenburg, M. Ashouei, J. Huisken, H. de Groot, V. van der Leest, G. J. Schrijen, M. van Hulst, and P. Tuyls, "Evaluation of 90nm 6T-SRAM as Physical Unclonable Function for secure key generation in wireless sensor nodes", in *2011 IEEE International Symposium of Circuits and Systems (ISCAS)*, May 2011, pp. 567–570. DOI: 10.1109/ISCAS.2011.5937628.

[76] G.-J. Schrijen and V. van der Leest, "Comparative Analysis of SRAM Memories Used As PUF Primitives", in *Proceedings of the Conference on Design, Automation and Test in Europe*, ser. DATE '12, Dresden, Germany: EDA Consortium, 2012, pp. 1319–1324, ISBN: 978-3-9810801-8-6. available from: `http://dl.acm.org/citation.cfm?id=2492708.2493033`.

[77] T. Ignatenko, G.-J. Schrijen, B. Skoric, P. Tuyls, and F. Willems, "Estimating the secrecy-rate of physical unclonable functions with the context-tree weighting method", in *2006 IEEE International Symposium on Information Theory*, IEEE, 2006, pp. 499–503.

[78] D. E. Holcomb, W. P. Burleson, and K. Fu, "Power-Up SRAM State as an Identifying Fingerprint and Source of True Random Numbers", *IEEE Transactions on Computers*, vol. 58, no. 9, pp. 1198–1210, Sep. 2009, ISSN: 1557-9956. DOI: 10.1109/TC.2008.212.

[79] C. Böhm, M. Hofer, and W. Pribyl, “A microcontroller SRAM-PUF”, in *Network and System Security (NSS), 2011 5th International Conference on*, Sep. 2011, pp. 269–273. DOI: 10.1109/ICNSS.2011.6060013.

[80] P. Tuyls, G.-J. Schrijen, B. Škoric, J. van Geloven, N. Verhaegh, and R. Wolters, “Read-Proof Hardware from Protective Coatings”, in *CHES 2006*, L. Goubin and M. Matsui, Eds., ser. LNCS, vol. 4249, Springer, 2006, pp. 369–383.

[81] G. Kömürcü and G. Dündar, “Determining the quality metrics for PUFs and performance evaluation of Two RO-PUFs”, in *10th IEEE International NEWCAS Conference*, Jun. 2012, pp. 73–76. DOI: 10.1109/NEWCAS.2012.6328959.

[82] R. Maes, “Physically unclonable functions: Constructions, properties and applications”, PhD thesis, KU Leuven, 2012.

[83] R. Maes, “An accurate probabilistic reliability model for silicon PUFs”, in *International Conference on Cryptographic Hardware and Embedded Systems*, Springer, 2013, pp. 73–89.

[84] Y. Dodis, R. Ostrovsky, L. Reyzin, and A. Smith, “Fuzzy extractors: How to generate strong keys from biometrics and other noisy data”, *SIAM Journal on Computing*, vol. 38, no. 1, pp. 97–139, 2008.

[**85**] F. Wilde, C. Frisch, and M. Pehl, “Efficient Bound for Conditional Min-Entropy of Physical Unclonable Functions Beyond IID”, in *2019 IEEE International Workshop on Information Forensics and Security (WIFS)*, Dec. 10, 2019, pp. 1–6. DOI: 10.1109/WIFS47025.2019.9035098.

[**86**] R. Hesselbarth, F. Wilde, C. Gu, and N. Hanley, “Large scale RO PUF analysis over slice type, evaluation time and temperature on 28nm Xilinx FPGAs”, in *2018 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, Apr. 2018, pp. 126–133. DOI: 10.1109/HST.2018.8383900.

[87] M. Pehl, M. Hiller, H. Graeb, M. Pehl, M. Hiller, and H. Graeb, “Efficient Evaluation of Physical Unclonable Functions Using Entropy Measures”, en, *Journal of Circuits, Systems and Computers*, vol. 25, no. 01 (2016) 1640001, p. 23, Oct. 2016. DOI: 10.1142/S0218126616400016. eprint: `https://doi.org/10.1142/S0218126616400016`. available from: `https://doi.org/10.1142/S0218126616400016`.

[88] C. Dietz, “Sensitivitätsanalyse zu PUF Metriken”, undergraduate internship supervised by myself, TUM, 2018.

[89] M. Bhargava and K. Mai, “A high reliability PUF using hot carrier injection based response reinforcement”, in *Cryptographic Hardware and Embedded Systems-CHES 2013*, Springer, 2013, pp. 90–106.

[90] J. L. Teugels, “Some representations of the multivariate Bernoulli and binomial distributions”, *Journal of Multivariate Analysis*, vol. 32, pp. 256–268, 2 1990. DOI: 10.1016/0047-259X(90)90084-U.

[**91**] F. Wilde and M. Pehl, “On the Confidence in Bit-Alias Measurement of Physical Unclonable Functions”, in *2019 17th IEEE International New Circuits and Systems Conference (NEWCAS)*, Jun. 2019, pp. 1–4. DOI: 10.1109/NEWCAS44328.2019.8961298.

[92] A. Agresti and B. A. Coull, “Approximate Is Better than "Exact" for Interval Estimation of Binomial Proportions”, *The American Statistician*, vol. 52, no. 2, pp. 119–126, 1998. DOI: 10.2307/2685469. available from: `http://www.jstor.org/stable/2685469`.

[93] M. Radev, “Using PAQ for entropy estimation of SRAM PUF data”, undergraduate internship supervised by myself, TUM, 2020.

[94] M. Mahoney, *The PAQ Data Compression Programs*, [Online, accessed 23-November-2020]. available from: `https://cs.fit.edu/mmahoney/compression/paq.html`.

[95] B. Knoll, *cmix*, [Online, accessed 23-November-2020]. available from: `http://www.byronknoll.com/cmix.html`.

[96] P. Deutsch, “DEFLATE Compressed Data Format Specification version 1.3”, RFC 1951, May 1996. DOI: 10.17487/RFC1951.

[97] F. M. J. Willems, Y. M. Shtarkov, and T. J. Tjalkens, “The context-tree weighting method: basic properties”, *IEEE transactions on information theory*, vol. 41, no. 3, pp. 653–664, 1995.

[98] M. Mahoney, *Large Text Compression Benchmark*, [Online, accessed 23-November-2020]. available from: `http://www.mattmahoney.net/dc/text.html`.

[99] S. U. Hussain, M. Majzoobi, and F. Koushanfar, “A built-in-self-test scheme for online evaluation of physical unclonable functions and true random number generators”, *IEEE Transactions on Multi-Scale Computing Systems*, vol. 2, no. 1, pp. 2–16, 2016.

[100] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, A. Heckert, J. Dray, and S. Vo, *A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications*, SP 800-22, 1a. NIST, Apr. 27, 2010. available from: `https://doi.org/10.6028/NIST.SP.800-22r1a`.

[101] W. Killmann and W. Schindler, *A proposal for: Functionality classes for random number generators*. Bundesamt für Sicherheit in der Informationstechnik, 2011. available from: `https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Zertifizierung/Interpretationen/AIS_31_Functionality_classes_for_random_number_generators_e.pdf?__blob=publicationFile&v=1`.

[102] M. D. Springer, *The algebra of random variables*. Wiley, 1979, ISBN: 0471014060.

[103] P. A. P. Moran, "Notes on continuous stochastic phenomena", *Biometrika*, vol. 37, no. 1/2, pp. 17–23, 1950.

[104] R. C. Geary, "The contiguity ratio and statistical mapping", *The incorporated statistician*, vol. 5, no. 3, pp. 115–146, 1954.

[105] P. A. P. Moran, "The Interpretation of Statistical Maps", *Journal Royal Statistical Society, Series B*, vol. 10, pp. 243–251, 1948.

[106] A. D. Cliff and J. K. Ord, *Spatial Autocorrelation*. Pion, London, 1973, ISBN 0850860369.

[**107**] *PUF Quality Assessment Suite (PQAS), Matlab Edition.* available from: `https://gitlab.lrz.de/tueisec/PQAS`.

[108] F. Daxer, "Implementation of a Built-in Self Test for PUFs", Bachelor's thesis supervised by myself, TUM, 2019.

[109] C. Glowacz, V. Grosso, R. Poussier, J. Schüth, and F.-X. Standaert, "Simpler and More Efficient Rank Estimation for Side-Channel Security Assessment", in *Fast Software Encryption*, G. Leander, Ed., Springer Berlin Heidelberg, 2015, pp. 117–129, ISBN: 978-3-662-48116-5. available from: `https://link.springer.com/chapter/10.1007%2F978-3-662-48116-5_6`.

[**110**] F. Wilde, B. Gammel, and M. Pehl, "Spatial Correlations in Physical Unclonable Functions", in *6th Conference on Trustworthy Manufacturing and Utilization of Secure Devices (TRUDEVICE 2016)*, UPC, Barcelona, Spain: UPC, 2016.

[111] N. Jacob, J. Wittmann, J. Heyszl, R. Hesselbarth, F. Wilde, M. Pehl, G. Sigl, and K. Fischer, "Securing FPGA SoC configurations independent of their manufacturers", in *2017 30th IEEE International System-on-Chip Conference (SOCC)*, Sep. 2017, pp. 114–119. DOI: 10.1109/SOCC.2017.8226019.

A. List of Publications

All publications appeared in peer-reviewed proceedings or journals.

A.1. Journals

[60] F. Wilde, B. M. Gammel, and M. Pehl, “Spatial Correlation Analysis on Physical Unclonable Functions”, *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 6, pp. 1468–1480, Jun. 2018, ISSN: 1556-6013. DOI: 10.1109/TIFS.2018.2791341

A.2. Conference Proceedings

[71] F. Wilde, M. Hiller, and M. Pehl, “Statistic-based security analysis of ring oscillator PUFs”, in *Integrated Circuits (ISIC), 2014 14th International Symposium on*, IEEE, 2014, pp. 148–151

[37] M. Pehl *et al.*, “Qualitätsevaluierung von Physical Unclonable Functions als Schlüsselspeicher”, de, in *14. Deutscher IT-Sicherheitskongress*, Bonn, Deutschland, May 2015

[110] F. Wilde, B. Gammel, and M. Pehl, “Spatial Correlations in Physical Unclonable Functions”, in *6th Conference on Trustworthy Manufacturing and Utilization of Secure Devices (TRUDEVICE 2016)*, UPC, Barcelona, Spain: UPC, 2016

[74] F. Wilde, “Large Scale Characterization of SRAM on Infineon XMC Microcontrollers as PUF”, en, in *4th Workshop on Cryptography and Security in Computing Systems (CS2 2017) HIPEAC17*, Stockholm, Sweden, Jan. 2017. DOI: 10.1145/3031836.3031839. available from: `https://dl.acm.org/citation.cfm?id=3031839`

[111] N. Jacob *et al.*, “Securing FPGA SoC configurations independent of their manufacturers”, in *2017 30th IEEE International System-on-Chip Conference (SOCC)*, Sep. 2017, pp. 114–119. DOI: 10.1109/SOCC.2017.8226019

[86] R. Hesselbarth *et al.*, “Large scale RO PUF analysis over slice type, evaluation time and temperature on 28nm Xilinx FPGAs”, in *2018 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, Apr. 2018, pp. 126–133. DOI: 10.1109/HST.2018.8383900

[91] F. Wilde and M. Pehl, “On the Confidence in Bit-Alias Measurement of Physical Unclonable Functions”, in *2019 17th IEEE International New Circuits and Systems Conference (NEWCAS)*, Jun. 2019, pp. 1–4. DOI: 10.1109/NEWCAS44328.2019.8961298

[85] F. Wilde, C. Frisch, and M. Pehl, “Efficient Bound for Conditional Min-Entropy of Physical Unclonable Functions Beyond IID”, in *2019 IEEE International Workshop on Information Forensics and Security (WIFS)*, Dec. 10, 2019, pp. 1–6. DOI: 10.1109/WIFS47025.2019.9035098