
A FEW SELECTED TOPICS ON BOOLEAN FUNCTIONS

Submitted to Indian Statistical Institute
in partial fulfillment of the thesis requirements for the Degree of
DOCTOR OF PHILOSOPHY IN COMPUTER SCIENCE



Author: **Animesh Roy**
Senior Research Fellow

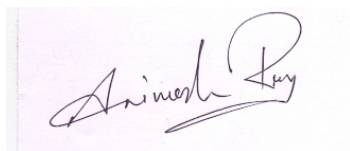
Supervisor: **Prof. Subhamoy Maitra**

Applied Statistics Unit
Indian Statistical Institute
Kolkata - 700108, India

To my family

DECLARATION OF AUTHORSHIP

I, **Animesh Roy**, a research scholar at the Applied Statistics Unit (ASU), Indian Statistical Institute (ISI), Kolkata, hereby declare that this thesis contains the research I conducted under the supervision of **Prof. Subhamoy Maitra** (ASU, ISI Kolkata). I affirm that the research and writing in this work are entirely my own original creation. To the best of my knowledge, none of the material included in this thesis has been previously published or authored by another individual, nor has it been submitted in whole or in part for any degree, diploma, or other academic qualification at any other institution.

A handwritten signature in black ink on a light pink background. The signature is written in a cursive style and reads "Animesh Roy".

Animesh Roy

Applied Statistics Unit

Indian Statistical Institute, Kolkata

203, Barrackpore Trunk Road

Kolkata 700108, INDIA.

ACKNOWLEDGEMENT

I want to seize this moment to convey my sincere thanks and deep appreciation to everyone who played a role in the completion of my thesis. Their assistance, direction, and motivation have been immensely valuable on this path.

I am delighted to express my sincere thanks to Prof. Subhamoy Maitra, my supervisor throughout my academic journey at Indian Statistical Institute (ISI). His consistent support and encouragement have been incredibly valuable to me. I am truly grateful for the chance to learn and grow under his guidance, which has played a vital role in reaching this important milestone.

I am immensely grateful to Baba, Maa and my sister Simpi for their unwavering support, encouragement and sacrifices throughout my academic journey. Their love, guidance and belief in me have been the pillars of strength that propelled me forward.

I am also grateful to my wife Mrs. Tanusree Roy for her constant support and cooperation at the final stage of this journey.

My sincere gratitude goes to the Applied Statistics Unit at the Indian Statistical Institute, Kolkata, for providing me with the necessary research facilities.

I am additionally thankful to Dr. Dibyendu Roy for his outstanding support throughout my PhD journey. He rigorously reviewed existing results in our field, identified significant problems and guided me in the right direction. I am also grateful to Dr. Arpita Maitra, Dr. Anirban Ghatak, and Prof. Pantelimon Stănică for their exceptional support and invaluable cooperation.

I would also like to express my heartfelt appreciation to Jyotirmoy, Suman, Subhra, Abhinav and Aniruddha Da for their expertise and constructive feedback that have shaped my research and enriched my academic growth. I would also like to thank Gourab, Subha, Chandranan, Subhadip Da, Susanta Da, Samir Da, Debendra, Rakesh, Anup, Diptendu Da, Avishek Da, Soumya, Koustabh and Debasmita for their guidance, inspiration, entertaining talks and mental exchanges. Also I am glad to have seniors like Avijit Da, Nilanjan Da, Laltu Da, Avik Da, Mustaf Da, Soumya Da, Partha Da.

I would also like to express my heartfelt appreciation to my dear friends Partha, Sourav, Ayan, Gopal, Souvik and Papan for their friendship, conversation, support and encouragement. I am very fortunate to have brothers Tutun Da, Bumba Da, Munaai Da, Avrojit, Avijit, Soham, Ankit, Joy and sisters Anindita, Piyali, Anushua.

Our gatherings, endless conversations and support have illuminated my path and made this journey smooth and enjoyable.

I am also thankful to my other family members for their continuous support, love and help. I wish to extend my heartfelt apologies to those individuals whose names may not be expressly mentioned here, yet who have offered invaluable support and cooperation, playing a significant role in the successful completion of my endeavors. Your contributions are profoundly valued and appreciated.

Thank you all for being an integral part of this remarkable journey.

LIST OF PUBLICATIONS/MANUSCRIPTS

Conferences

1. **A. Roy**, D. Roy, and S. Maitra. How Do the Arbiter PUFs Sample the Boolean Function Class? In *Selected Areas in Cryptography - 28th International Conference, SAC 2021, Virtual Event, September 29 - October 1, 2021, Revised Selected Papers*, volume 13203 of *Lecture Notes in Computer Science*, pages 111–130. Springer, 2021.
doi: https://doi.org/10.1007/978-3-030-99277-4_6
2. B. Chatterjee, R. Parikh, A. Maitra, S. Maitra and **A. Roy**. Revisiting BoolTest - On Randomness Testing Using Boolean Functions. In *Progress in Cryptology – INDOCRYPT 2022*: 471-491.
doi: https://doi.org/10.1007/978-3-031-22912-1_21

Journals

3. J. Basak, S. Maitra, P. Paul and **A. Roy**. Analysis of Boolean Functions related to binary input binary output two-party nonlocal games. (Accepted for presentation at BFA 2022), In *Cryptography and Communications*, 15(5): 861-890 (2023),
doi: <https://doi.org/10.1007/S12095-023-00648-0>
4. M. Kansal, **A. Roy**, D. Roy, S. Bodapati and A. Chattopadhyay. Priority Arbiter PUF: Analysis. In *Discret. Appl. Math.* 356: 71-95 (2024).
doi: <https://doi.org/10.1016/j.dam.2024.05.013>

Preprints

5. **A. Roy**, D. Roy and P. Stănică. On Combining Arbiter based PUFs. Communicated to: *Cryptography and Communications*, submitted on 17/02/2024. Submission ID: 40d69b49-d6ed-42c9-a658-760cffc1319e (1st round review received. Reviewers suggested some minor comments. Will submit the revised version very soon.)

ABSTRACT

In this thesis, we use the techniques of Boolean functions in different applications. More specifically, our focus is on the properties of Boolean functions that hold cryptographic significance. The employed techniques primarily revolve around combinatorial methods, yielding fresh insights into the enumeration and construction of such functions.

Initially, our effort is on exploring functions generated through an Arbiter-based Physically Unclonable Function (PUF) construction with random delay parameters. We observe that, under specific constraints on input weights, such a straightforward model of Arbiter PUFs yields favourable cryptographic parameters in terms of differential analysis. In this context, we theoretically address the autocorrelation properties issue within a restricted space of input variables with fixed weights.

While investigating this aspect independently, we have concentrated on the connection between Arbiter PUFs and Threshold functions. Subsequently, we delve into the study of the combination of such Arbiter-based PUFs, specifically examining the scenario involving the XOR operation of two devices with arbitrary inputs of equal length. Based upon extensive computations, we identify several interesting properties. Beyond addressing the counting problem, we also derive the general formula to calculate the probability of the occurrence of identical outputs from a combiner model PUF corresponding to two distinct challenge inputs.

We further note that the collection of Boolean functions produced by Priority Arbiter PUF (PA-PUF) is larger than the set of Boolean functions generated by the traditional Arbiter-based PUFs. Lastly, we investigate the bias estimation of the response bit generated by PA-PUF concerning the influence of altering specific bits in the challenge input.

Next, we assess whether a seemingly random stream exhibits any underlying non-random patterns. In this context, we highlight specific constraints of the *BoolTest* strategy proposed by Sýs et al (2017) and introduce combinatorial findings related to the identification of optimal Boolean functions in this regard. Our results identify the most effective Boolean function in this context, one that yields the maximum Z -score.

Subsequently, we conducted a thorough examination of all Boolean functions involving four variables to formulate binary input-binary output two-party nonlocal

games and assess their efficacy in both classical and quantum contexts. Our investigation identifies certain games other than the CHSH game (which is naturally the provably best example) that exhibit a better (may be sub-optimal compared to the best case) success probability in quantum scenarios compared to classical scenarios. Additionally, we extend the framework of the classical strategies to encompass other n -party nonlocal games.

Contents

1	Introduction	21
1.1	Thesis Plan	28
1.2	Prerequisites	30
2	Background	31
2.1	Basics of Boolean functions	31
2.1.1	Restricted Domain	32
2.2	Arbiter PUF as a Boolean function	33
2.3	Modeling XOR of Arbiter-based PUFs as Boolean functions	35
2.3.1	XOR-PUF from the same length Arbiter PUFs	36
2.3.2	XOR-PUF from different lengths Arbiter PUFs	36
2.4	Modeling Priority Arbiter-based PUF as Boolean functions	37
2.5	Evaluation of Randomness through Boolean functions	42
2.5.1	Brief Description of <i>BoolTest</i> by Sýs et al [SKŠ17]	44
2.6	Two-Party Nonlocal Games in terms of Boolean functions	47
3	How do the Arbiter PUFs Sample the Boolean Function Class?	51
3.1	Introduction	52
3.2	Motivation	52
3.3	Relation Between $\mathcal{B}_n^{\text{PUF}}$ and \mathcal{B}_n	54
3.4	Determining whether $f \in \mathcal{B}_n^{\text{PUF}}$	65
3.4.1	Improved but randomized technique	68

3.5	On Restricted Autocorrelation of Arbiter PUF	70
3.5.1	Theoretical Analysis	71
3.6	Conclusion	74
4	On Combining Arbiter-based PUFs	77
4.1	Introduction	78
4.2	Understanding the relation between $\mathcal{B}_n^{\text{XOR-PUF}}$, $\mathcal{B}_{n,m}^{\text{XOR-PUF}}$ and \mathcal{B}_n . . .	79
4.2.1	The relation between $\mathcal{B}_{n,m}^{\text{XOR-PUF}}$ and $\mathcal{B}_n^{\text{PUF}}$	79
4.2.2	The relation between $\mathcal{B}_n^{\text{XOR-PUF}}$ and $\mathcal{B}_n^{\text{PUF}}$	81
4.3	Theoretical estimation of bias in PUF with combiner function	86
4.4	Conclusion	89
5	Priority Arbiter PUF: Analysis	91
5.1	Introduction	92
5.2	Motivation	93
5.3	Characterization of $\mathcal{B}_n^{\text{PA-PUF}}$	93
5.4	Theoretical Estimation of Bias in PA-PUF	99
5.5	Experimental Results and Analysis	113
5.5.1	Intra-Chip Hamming Distance	114
5.5.2	Uniformity	115
5.5.3	Reliability	115
5.5.4	Uniqueness	116
5.5.5	Bit-Aliasing	118
5.5.6	Comparisons	119
5.5.7	Machine Learning-based Modeling Attacks	119
5.6	Conclusion	120
6	Revisiting <i>BoolTest</i> – On Randomness testing using Boolean functions	123
6.1	Introduction	124
6.2	Critical evaluations of <i>Z</i> -score	124
6.2.1	<i>Z</i> -score for data with all and equal frequency inputs	124
6.2.2	Maximum <i>Z</i> -score for frequencies s and $s + 1$	125

6.2.3	Maximum Z -score when some of the patterns arrive only, and only once	127
6.3	Finding the best Boolean function to have maximum Z -score	127
6.3.1	Improving the time and space complexity further	132
6.4	Results	136
6.4.1	RC4	136
6.4.2	Comparison with Java rand and AES [DR02, AES01]	137
6.4.3	Cross-testing by the generated polynomials, i.e., functions	138
6.5	Conclusion	139
7	Analysis of Boolean Functions Related to Two-party Nonlocal Games	143
7.1	Introduction	144
7.2	Inconsistency for the $2 + 2$ Partition and Its Subpartitions	145
7.3	Results related to classical strategies for any n -party nonlocal game	147
7.4	Some Basic Results	149
7.5	Analysis of the Maximum Success Probability in Classical Scenario	159
7.6	Analysis of the Maximum Success Probability in Quantum Scenario	160
7.7	Analysis of the Results in [DPM19]	163
7.8	Analysis of the Binary Input Binary Output Two-party Nonlocal Games	165
7.8.1	Games Corresponding to 8 Successful Outcomes	166
7.8.2	Games Corresponding to 9 Successful Outcomes	172
7.8.3	Games Corresponding to 10 Successful Outcomes	174
7.8.4	Games Corresponding to 11 Successful Outcomes	175
7.8.5	Games Corresponding to 12 or More Successful Outcomes	176
7.8.6	Games Corresponding to 7 Successful Outcomes	176
7.8.7	Games Corresponding to 6 Successful Outcomes	178
7.8.8	Games Corresponding to 5 Successful Outcomes	180
7.8.9	Games Corresponding to 4 Successful Outcomes	182
7.8.10	Games Corresponding to 3 or Less Successful Outcomes	183
7.9	Conclusion	183
8	Conclusion	185
8.1	Brief Summary & Further Research Directions	185

List of Figures

1-1	Ring Oscillator PUF	22
1-2	Arbiter PUF with a Feed Forward Arbiter in the data path; Challenges are considered as 0/1 instead of 1/-1.	25
2-1	Basic structure of an Arbiter based PUF	33
2-2	Dealy segment notation in each stage	34
2-3	XOR-PUF from same length Arbiter PUFs having same inputs	37
2-4	XOR-PUF from different length Arbiter PUFs	37
2-5	i -th stage of PA-PUF with three paths	39
3-1	Representation of Table 3.1	53
3-2	Comparison of Autocorrelation Distribution in Complete Domain	54
3-3	Comparison of the restricted autocorrelation.	70
3-4	Distribution of $Pr[z_C = z_{\bar{C}}]$ in $E_{12,k}$	71
5-1	Schematic of proposed PA-PUF which includes three parallel multiplexer lines (T, C, and B) and a priority arbiter at the end to generate the response bit. Challenges are considered as 0/1 instead of 1/-1.	92
5-2	Comparison of the restricted autocorrelation.	100
5-3	Comparison of $Pr[z_C = z_{\bar{C}}]$ between PUF and PA-PUF for different flip locations.	113
5-4	Intra-Chip Hamming Distance of the PA-PUF (a) 64-bit Response (b)128-bit Response.	114

5-5	Responses of the PA-PUF to show uniformity (a)64-bit PA-PUF (b)128-bit PA-PUF.	115
5-6	Uniformity of the PA-PUF (a) 64-bit Response (b)128-bit Response. .	116
5-7	Response of the PA-PUF to show reliability (a) 64-bit Response (b)128-bit Response.	117
5-8	Reliability of the PA-PUF (a) 64-bit Response (b)128-bit Response. .	117
5-9	Reliability of the PA-PUF using BCH error correction codes; Resulting reliability of 100%.	118
5-10	Inter-Chip Hamming Distance of the PA-PUF (a) 64-bit Response (b)128-bit Response.	119
5-11	Learning Accuracy for Various Machine Learning Attacks on the PA-PUF	120

List of Tables

3.1	Experimental Bias of PUFs ($n = 12$) in complete domain (over 1024 randomly chosen Arbiter PUFs) for single bit difference, matching with the theoretical values from [SBC ⁺ 19b]	53
3.2	$ \mathcal{B}_n^{\text{PUF}} $ for different n	63
3.3	Experimental validation of Algorithm 3. The functions are written in hexadecimal format.	67
3.4	$Pr[z_C = z_{\tilde{C}}]$ in $E_{12,k}$	71
4.1	$ \mathcal{B}_n^{\text{XOR-PUF}} $ for different n	84
4.2	$ \mathcal{B}_n^{\text{g(d)}} $ for different $n \leq 4$	86
5.1	Comparison between $\mathcal{B}_n^{\text{PA-PUF}}$ and $\mathcal{B}_n^{\text{PUF}}$.	97
5.2	Experimental validation of Algorithm 4.	98
5.3	Experimental validation of Theorem 11	107
5.4	Experimental validation of Theorem 12	112
5.5	Comparison of $Pr[z_C = z_{\tilde{C}}]$ between PUF and PA-PUF of length 11.	113
5.6	Comparison of the performance metrics of the ring oscillator PUF ; Ideal value of reliability is 100% while the remaining parameters have an ideal value of 50%.	119
5.7	Comparisons of different PUF designs; ‘U’ is the uniqueness and ‘R’ is the reliability.	120

5.8	Prediction Accuracy of Various Arbiter-based PUF. (lower prediction accuracy represents the PUF has more robustness against modeling attacks.)	121
6.1	Testing RC4 2nd byte samples.	137
6.2	Results	137
6.3	Cross-testing with <i>BoolTest</i> [SKŠ17].	138
6.4	Cross-testing for our algorithm.	139
7.1	Inconsistent Outputs	146
7.2	Success probabilities of a game for all possible classical strategies	161
7.3	Analysis of partitions for 8 successful outcomes	171
7.4	Analysis of partitions for 9 successful outcomes	172
7.5	Analysis of partitions for 10 successful outcomes	174
7.6	Analysis of partitions for 11 successful outcomes	176
7.7	Analysis of partitions for 7 successful outcomes	177
7.8	Analysis of partitions for 6 successful outcomes	179
7.9	Analysis of partitions for 5 successful outcomes	180
7.10	Analysis of partition for 4 successful outcomes	182
7.11	List of partitions and the corresponding nonlocal games (in ANF form) which offer quantum advantage.	183

LIST OF ACROYNMS AND ABBREVIATIONS

Expansion	Acronyms/ Abbreviations
Algebraic Normal Form	ANF
Physical Unclonable Function	PUF
Priority Arbiter-based PUF	PA-PUF
Exclusive-OR	XOR
Most Significant Bit	MSB
Clauser-Horne-Shimony-Holt	CHSH
Greenberger-Horne-Zeilinger	GHZ
That is	i.e.
Without loss of generality	WLOG
Bit-wise XOR	(\wedge)

LIST OF SYMBOLS

In this thesis, we outline all the symbols and notations employed herein.

- \mathbf{x}, \mathbf{a} : Vectors are written in bold lower case.
- $\Pr(A)$: Probability of occurrence of an event A .
- $\mathcal{N}(\mu, \sigma)$: Denotes normal distribution with mean μ and standard deviation σ .
- $wt(\mathbf{a})$: Number of 1's in \mathbf{a} .
- \mathcal{B}_n : Denotes the collection of all n -variable boolean functions.
- $\mathcal{B}_n^{\text{PUF}}$: Denotes the collection of all n -variable Boolean functions generated from an n -length Arbiter based PUF.
- $\mathcal{B}_n^{\text{XOR-PUF}}$: Denotes the collection of all n -variable Boolean functions generated from XOR of two n -length Arbiter based PUFs.
- $\mathcal{B}_{n,m}^{\text{XOR-PUF}}$: Denotes the collection of all n -variable Boolean functions generated from XOR of two Arbiter based PUFs of length n and m .
- $\mathcal{B}_n^{k\text{-XOR-PUF}}$: Denotes the collection of all n -variable Boolean functions generated from the XOR of k many n -length Arbiter PUFs.
- $E_{n,k}$: Denotes the set of all n -length strings whose weight is k .
- $\mathcal{A}_f(\mathbf{a})$: Denotes autocorrelation of an n -variable Boolean function f at $\mathbf{a} \in \{0, 1\}^n$.
- $\mathcal{A}_f^{E_{n,k}}$: Restricted autocorrelation of f over $E_{n,k}$.

Contents

1.1 Thesis Plan	28
1.2 Prerequisites	30

In this thesis we investigate various properties of Boolean functions in the context of classical and quantum cryptographic primitives. The first part of the thesis deals with the analysis and construction of Physically Unclonable Functions (PUF), while the second part contains our efforts in the testing of randomness of Boolean functions and quantum advantages related to nonlocal games. In the two major parts of the thesis, the unifying thread of Boolean functions weaves various topics involving the broad area of cryptology.

A Physically Unclonable Function or PUF is a technical umbrella-term for any physical entity whose operation cannot be replicated/“cloned” by means of another system with identical technology. For specified input conditions, or challenges in cryptographic terms, a PUF provides a physically defined unique identifier. For instance, PUFs are used in many hardware devices (e.g., RFID tags and smart cards) to generate cryptographic keys, to achieve secure device authentication and to ensure various aspects of secure communication [LLG⁺04]. Integrated circuits are prone to random manufacturing variations such as process, voltage, and temperature, which may result in the variations of circuit delay. A PUF generates a unique response from a device for each challenge by exploiting such variations, thereby creating a unique Challenge-Response Pair (CRP) for each PUF [LLG⁺04]. Theoretical and experimental investigation of CRP set expansion has been demonstrated in [CCKM16]. The

unique challenge-response pairs are utilized for various security protocols, such as device authentication and data encryption [CCM17, GCL⁺21, RDM⁺23, MCNS15, Muk16, CGS⁺19].

Another important role of the PUFs is to generate random-looking keys for cryptographic computations in run-time without storing those secret patterns in non-volatile memory. As such the outputs from a PUF should meet certain requirements, mostly in terms of randomness. However, it must be understood that once several PUFs are fabricated from the same family, each of them is fixed and should be different. That is, it is desirable that the truth tables of two different PUFs from the same family of fabrication will have significantly different patterns. Another important consideration is that, a PUF being a hardware-based bit generator, there can be the presence of noise, mostly environmental, in the output bits. In [Gas03], it has been proposed that error-correcting codes can be used in such scenarios.

PUFs can be classified into delay-based and memory-based categories based on the parameters from which their responses are derived. A delay-based PUF uses the difference in delay between two paths to derive the response. Prominent examples of such delay-based PUFs are the Arbiter-based PUF [LLG⁺05] and the Ring Oscillator PUF [SD07]. A semiconductor memory, once it is turned on, will store random data in its memory cells. The random data from the memory can be used to generate a PUF response - this is the principle of memory-based PUFs such as the SRAM PUF [GKST07], the Butterfly PUF [KGM⁺08], the Glitch PUF [SS10] and the MEemory Cell-based Chip Authentication (MECCA) PUF [KNWB11].

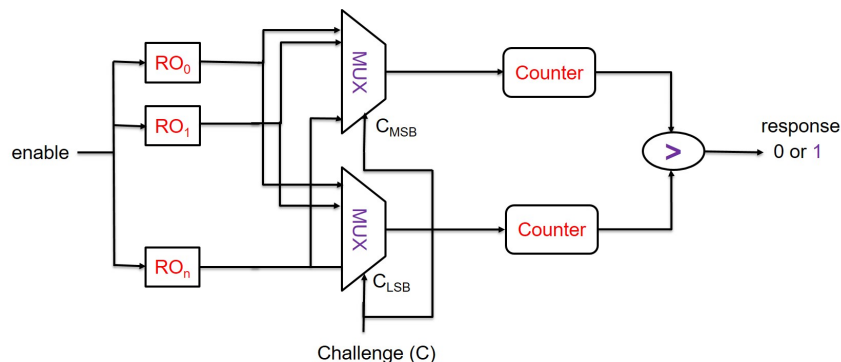


Figure 1-1: Ring Oscillator PUF

A Ring Oscillator is an electronic circuit to generate square waves. In a Ring

Oscillator PUF [SD07], an n number of ring oscillators were used to generate n square waves of a similar frequency, as shown in Figure 1-1. Due to the diverse paths through which oscillation outputs propagate, the skew of the square wave fluctuates, resulting in the generation of a response. A random challenge selects two of the n square waves and tallies the total number of rising or falling edges of the two square waves, designated as top and bottom counters. A comparator decides the response as ‘1’ (or ‘0’) depending upon whether the top counter is greater than the bottom counter (or vice versa). An extension to the Ring Oscillator has been proposed, such as the Configurable Ring Oscillator PUF [SVCL18], which uses a linear feedback shift register to generate new challenges for the PUF.

In our thesis, we focus on Arbiter-based PUFs and their variants. An Arbiter-based PUF (Arbiter PUFs) is a hardware-based pseudorandom bit generator that was introduced by Gassend et al. [GCDD02], [PRTG02] in the year 2002. Such properties are achievable (to some extent) as the output bit from any Arbiter PUF depends on device-specific parameters such as delay parameters (defined later). One may assume that these delay parameters are identically independent random variables which follow a normal distribution. Such random parameters differentiate each instance from the same family of PUFs. An n -length Arbiter PUF takes input (also called challenge input from a cryptanalytic point of view) from $\{-1, 1\}^n$ (for mathematical analysis, we use $\{-1, 1\}^n$ instead of $\{0, 1\}^n$) and generates either 0 or 1. The circuit of the PUF is based on two paths where an input voltage moves from the first layer to the last layer via any of the two paths in each stage. One path is known as the top path, and the other is referred to as the bottom path. Due to the delay parameters, the arrival time of the voltage via the two paths will be different. The circuit of PUF measures the delay difference between these two paths, and depending upon the sign of delay difference, it generates as output either 0 or 1.

A Boolean function can be viewed as a mapping from $\{0, 1\}^n$ to $\{0, 1\}$. Thus, an n -length Arbiter PUF can act as an n -variable Boolean function. In most cryptographic applications, the input bits of a Boolean function are usually considered independent and taken uniformly from the domain and it is expected that the output bits would display an acceptable measure of pseudo-randomness. Hence the guarantee of pseudo-randomness in the output is necessary for a PUF model to be acceptable for cryptographic applications. In recent time, multiple articles [Bec15, BFSK11, RBK10,

[RDK12, RSS⁺10a, RSS09, RSS⁺13a] have analyzed varied models of PUFs and observed different weaknesses. Multiple machine learning-based tools have also been developed to analyze the security of different models of PUF [Lim05, MKP08, RSS⁺10a, RSS⁺13a, CCMC20]. In response, several counter-measures have been proposed to resist such attacks, leading to the introduction of new designs [Dev09, LLG⁺04, Lim05]. In a parallel development, we are also interested in the combinatorial and statistical aspects in evaluating the Boolean functions generated out of varying delays in Arbiter PUFs. In this context, we refer to [SBC⁺19b], where several non-randomness results have been demonstrated theoretically.

As an additional note, it’s worth mentioning the operational connection with specific stream ciphers such as FLIP [MJSC16], which serve as essential elements in Fully Homomorphic Encryption (FHE) [CCF⁺18]. FLIP belongs to a class of stream ciphers whose keystream bit is generated by using a non-linear Boolean function with inputs from a restricted domain. In this context, several properties of Boolean functions over the restricted domain (definition of the restricted domain is elaborated in Section 2.1.1) were studied in [CaR17, MMM⁺18, MMM⁺19, MZD19]. Our findings demonstrate that although there is significant bias in the autocorrelation measures of Arbiter PUFs [SBC⁺19b], this bias disappears when challenge inputs are selected from a restricted domain.

While investigating the number of Boolean functions generated by Arbiter PUFs, we had not explored the connection between Arbiter PUFs and Linear Threshold Function. In 1961, Muroga, Toda, and Takasu had introduced and explored the Majority Decision Function or Threshold Function [MTT61]. The total count of n -variable threshold functions precisely matches the number of Boolean functions derived from n -bit Arbiter PUFs. In Section 6.1.2 of [Sor17], it was noted that an Arbiter PUF could be represented as a threshold function. Considering this connection, we demonstrate methods for distinguishing Arbiter PUFs from other Boolean functions. Additionally, it illustrates how one can investigate a substantial portion of the Boolean functions class by XORing two n -length PUFs with identical inputs.

Research presented in [RRM21] has revealed that only a fraction of the entire set of Boolean functions can be produced using Arbiter PUFs. This prompts us to investigate the number of Boolean functions derivable from the circuitry of an XOR-PUF [RSS⁺10b]. In this study, we explore two variations of XOR-PUFs, one

with uniform input lengths and the other with disparate input lengths. Additionally, we examine the XORNF (Exclusive OR Normal Form) representation of two Arbiter PUFs of identical lengths, observing the resulting count of Boolean functions generated from them.

Various extensions to the basic Arbiter PUF exist to enhance the randomness of the response bit. One approach involves adjusting the path by introducing an Arbiter in the data path, as demonstrated in Figure 1-2, using a Feed Forward Arbiter PUF [LP14]. Other alternatives of Arbiter PUF are to increase the number of arbiters in the PUF circuit and combine the responses using XOR logic to generate a single response. These methods are proposed to enhance the uniqueness, reliability, and robustness of the PUF output, and such designs are multi-Arbiter PUF [KI15], [ZPK⁺16] and double Arbiter PUF [MYIS14], [MYIS15]. An alternative design is Programmable delay line-based Arbiter PUF [MKD10, AHT21, AHS22, AHS17]. An efficient implementation of FPGA-based XOR Arbiter PUF with significantly enhanced performances has been presented in [AHC22]. Since the non-linearity in the arbiter is relatively low, incorporating multi-arbiter designs can enhance the non-linearity in the overall design.

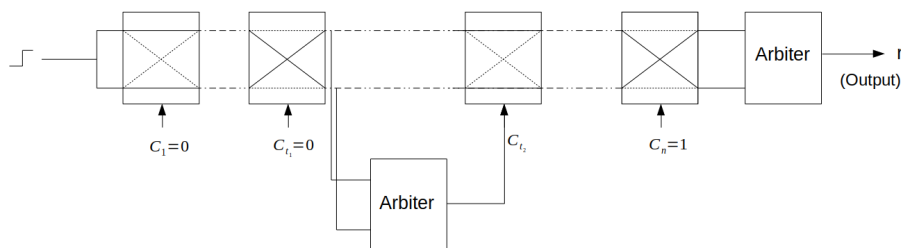


Figure 1-2: Arbiter PUF with a Feed Forward Arbiter in the data path; Challenges are considered as 0/1 instead of 1/-1.

A combination of Arbiter PUF and Ring Oscillator PUF is presented in [SSM⁺14]. The response of the PUF can be modelled mathematically to derive the complete set of challenge-response pairs. In literature, there are some designs of PUF which are vulnerable to machine learning attack [WMP⁺20], [RSS⁺13b], [TAB20], reliability attack [DV13], [Bec15]. Authors in [CCMH21] discuss the strict avalanche criterion (SAC) resistant delay-based PUF. Further, authors in [WTM⁺22] discuss the neural network modelling attacks on the Arbiter PUF.

Given the analytical findings and identified weaknesses of the Arbiter PUF uti-

lizing two paths, a natural enquiry arises: What implications arise from increasing the number of paths within the PUF? In essence, can an Arbiter PUF be devised with more than two paths? In 2022, Singh et al. [SBP⁺22] introduced a revised construction of the Arbiter PUF featuring three distinct paths (Top, Center and Bottom paths), termed the Priority Arbiter PUF (PA-PUF). While the authors conducted an experimental analysis demonstrating enhanced robustness in their proposed design, a thorough mathematical analysis was not conducted. In this thesis, we undertake a detailed examination of the PA-PUF.

In a different direction, next we focus on (pseudo) randomness testing using Boolean functions. The extensive array of security applications requires data that is either truly random or appears indistinguishable. Therefore, it is essential to scrutinize whether a seemingly random data stream exhibits any underlying non-random pattern. However, classical computers operate deterministically, making it impossible to generate true randomness from them beyond the influence of an initial random seed, if present. Although pseudo-random number generation is pivotal in cryptology and other information technology domains, it inherently lacks genuine randomness.

Given the necessity to analyze the security characteristics of a cipher, the presence of randomness holds significant importance. In the pursuit of cryptanalysis of a cipher, a crucial approach involves crafting specific distinguishers capable of revealing insights into the cipher's inadequate confusion and diffusion properties. This entails identifying the extent to which the cipher's output diverges from true randomness. Examining the algorithm of the cipher to derive such a distinguisher is naturally a more scientific approach. For example, the famous distinguisher [MS02] against RC4 could be identified from its algorithm. However, a complicated design will always make identifying such distinguishers harder. In this regard, applying statistical tests on the reduced rounds of the cipher might provide a quicker way to identify certain non-randomness, and a more formal design of a distinguisher may be initiated from that observation. For methodologies where the random numbers are generated from physical processes such as quantum mechanics or thermal processes, such statistical tests might be useful. Our focus, therefore, will not be to look at the algorithms to find the weaknesses but study the data and try to obtain some statistical measure that will possibly differentiate the available stream at hand from some ideal data generated from a random source. In this direction, one may refer to the well-defined statistical

test-beds like Diehard [Mar95], Dieharder [BEB], NIST SP 800-22 [BRS⁺10], CryptX [GDNC94] and ENT [Wal18].

All these test suites (often called a battery) generally consist of a series of empirical tests of randomness. Each test aims to find a predefined pattern of bits (or block of bits) in the data under consideration and examines the randomness property by certain measures that can be computed from the occurrences of the predefined bit patterns. Each test results in a distribution of a specific feature of bits or blocks of bits. The distribution is then statistically compared with the expected one obtained from random data. The data under examination is considered to be non-random if the distributions differ significantly.

In theory, it is possible to create an infinite number of tests for certifying randomness, each with its strengths and weaknesses. Thus, understanding each statistical test is crucial rather than using them as black boxes. In this context, our focus is on the *BoolTest* [SKŠ17]. Here, every block of the bit-stream is applied to a carefully selected Boolean function, and the resulting output bits are analyzed. In Chapter 6, we revisit the methods outlined in [SKŠ17], pinpoint certain constraints of the test and subsequently propose several techniques to enhance the approach.

In our concluding contribution in this thesis, we concentrate on Boolean functions associated with binary input-binary output two-party nonlocal games. Quantum computing demonstrates a significantly greater level of computational prowess compared to classical methods, as evidenced by the remarkable speed of quantum algorithms, which can outperform conventional classical algorithms by exponential margins [Sho94, Sim94]. The potency of quantum computation lends itself to offering additional security through quantum cryptography [BB14], a level of protection unattainable in classical systems. Moreover, this quantum advantage extends to nonlocal games, showcasing the unique capabilities of quantum computing in various contexts.

Nonlocal games are of interest because they have the potential to achieve a quantum advantage or a distinct separation (an advantage to the maximum quantum success probability as compared to the maximum classical one) in certain instances. This capability is often valuable for demonstrating the quantum nature of a system and for certifying the integrity of untrusted devices within a Device Independent (DI) [RUV13] scenario. However, the inputs and the outputs for any of those games

are not restricted to bits. To the best of our knowledge, the CHSH game is the only known binary input-binary output two-party nonlocal game that offers a quantum advantage.

A seminal contribution in this direction is the Bell inequality [Bel64], introduced by John Bell in 1964. Bell’s theorem established a ground-breaking result by demonstrating the existence of correlations in quantum systems that cannot be explained by local hidden variable theories. This pivotal work paved the way for the exploration of nonlocal games as a means to test and investigate the implications of quantum mechanics experimentally. Following Bell’s theorem, significant scholarly efforts have been dedicated to crafting and scrutinizing nonlocal games to explore quantum phenomena and their practical implications.

In Chapter 7 of this thesis, we analyze the performance of all those binary input binary output two-party nonlocal games (in both classical and quantum scenarios) which have at least one successful outcome for every possible input. Our analysis has resulted in the formulation of some games (other than CHSH game) that yield a higher success rate in quantum scenarios compared to classical ones, even though the CHSH provides optimal separation. Also we have extended the characterization of classical strategies to any n -party nonlocal game.

1.1 Thesis Plan

The thesis consists of five chapters. The thesis structure is briefly summarized as follows. In Chapter 2, we address the fundamental background material necessary for the following chapters. Chapter 3 to Chapter 7 consist of the main contributions of this thesis. Chapter 8 serves as the conclusion, where several interesting problems for future research are also addressed.

The initial chapter, Chapter 3, makes the first significant contribution. Here, we investigate the exploration of functions obtained through an Arbiter-based PUF construction with randomly selected delay parameters. Also we have analyzed the autocorrelation from a specific angle and discussed relevant discoveries in this area. It is common knowledge that Boolean functions derived from Arbiter PUFs demonstrate noticeable biases in their autocorrelation characteristics under certain conditions. Nevertheless, it’s important to highlight that these biases can be mitigated by

imposing restrictions on input weights and a specific input bit difference.

In Chapter 4, we consider the combination of Arbiter-based PUFs, specifically the XOR of two Arbiter PUFs. We observed that among all XORNF representations of two Arbiter PUFs, the XOR PUF yields a significant portion of the Boolean functions class. Through extensive computations, we consequently propose a conjecture. Finally, we focus on a general PUF combiner and calculate the likelihood of two outputs from the combiner model PUF, corresponding to distinct challenge inputs and matching.

Moreover, we have performed detailed analysis of the Priority Arbiter-based PUF in Chapter 5. We note that the set of Boolean functions generated from PA-PUF is notably larger than those generated from classical Arbiter-based PUF. Further, we look into the bias in the output bit of PA-PUF when a certain challenge input bit is flipped. In the final part of this chapter, we compare cryptographic properties of PA-PUF with the existing models of PUF and we observe that PA-PUF possesses better cryptographic characteristics like Uniformity, Reliability, Uniqueness etc.

Next we consider a different application related to randomness testing where Boolean functions are exploited. In Chapter 6, we have discussed various key aspects concerning *BoolTest* [SKŠ17], recognized as a method for assessing the randomness of data streams. We introduce combinatorial findings aimed at identifying the optimal Boolean functions for maximizing the Z -score, achieving a complexity of $O(N \log N)$ concerning the input stream's length N , which exceeds the capabilities of the heuristic proposed for *BoolTest* [SKŠ17].

As pointed out earlier, we have also studied certain properties of Boolean functions related to certain games and their analyses both in classical and quantum contexts. In Chapter 7 we have extensively investigated all Boolean functions with four variables to represent binary input-binary output two-party nonlocal games and analyze their performance in both classical and quantum settings. Our analysis reveals specific games, apart from the CHSH game, that demonstrate a greater likelihood of success in quantum scenarios compared to classical ones. We further expand the categorization of classical strategies to encompass any n -party nonlocal game.

1.2 Prerequisites

It is assumed that the reader is familiar with undergraduate level mathematics. We will present more details regarding the more involved algebraic and combinatorial structures in Chapter 2. Basic understanding of computer algorithms would be helpful to understand a few methodologies. There is no requirement to have any background on PUFs, Randomness testing or Quantum paradigm. We will develop the background with sufficient details in the following sections and introduce the ideas one by one, as and when required.

Contents

2.1	Basics of Boolean functions	31
2.2	Arbiter PUF as a Boolean function	33
2.3	Modeling XOR of Arbiter-based PUFs as Boolean functions	35
2.4	Modeling Priority Arbiter-based PUF as Boolean functions	37
2.5	Evaluation of Randomness through Boolean functions	42
2.6	Two-Party Nonlocal Games in terms of Boolean functions	47

This chapter aims to present fundamental definitions and essential tools that will be utilized throughout the thesis. We recommend that readers who are already acquainted with Boolean functions briefly revisit this chapter to acquaint themselves with the notation. Before proceeding further, let us take a moment to recall some fundamental facts regarding Boolean functions.

2.1 Basics of Boolean functions

An n -variable Boolean function is a mapping from $\{0, 1\}^n$ to $\{0, 1\}$. Any Boolean function f in n -variable has a unique polynomial representation, which is known as Algebraic Normal Form (ANF) of f , such that $f(\mathbf{x}) = \sum_{\mathbf{a} \in \{0, 1\}^n} \mu_{\mathbf{a}} (\prod_{i=1}^n x_i^{a_i})$, for all $\mathbf{x} = (x_1, x_2, \dots, x_n)$, $\mathbf{a} = (a_1, a_2, \dots, a_n) \in \{0, 1\}^n$ and $\mu_{\mathbf{a}} \in \{0, 1\}$. In the

truth table representation, the inputs of the form (x_1, x_2, \dots, x_n) from $\{0, 1\}^n$ are arranged in lexicographically increasing order. The algebraic degree of f is $\deg(f) = \max_{\mathbf{a} \in \{0, 1\}^n} \{wt(\mathbf{a}) : \mu_{\mathbf{a}} \neq 0\}$, the degree of highest degree term(s) with nonzero coefficient in its ANF. Here $wt(\mathbf{a})$ is the number of 1's in \mathbf{a} . Let \mathcal{B}_n be the collection of all n -variable Boolean functions. Then any function $f \in \mathcal{B}_n$ can be expressed as,

$$f = [f(0, 0, \dots, 0), f(0, 0, \dots, 1), \dots, f(1, 1, \dots, 1)]$$

2.1.1 Restricted Domain

Let f be a function from $\{-1, 1\}^n$ to $\{0, 1\}$. Further, let the function be defined over a restricted domain when it takes input from a subset of $\{-1, 1\}^n$. We know that the weight of $\mathbf{x} \in \{0, 1\}^n$ (i.e., $wt(\mathbf{x})$) is considered as the number of 1's present in \mathbf{x} . In the similar convention along with the transformation $a \rightarrow (-1)^a$ here we define $wt(\mathbf{x})$ for $\mathbf{x} \in \{-1, 1\}^n$. The weight of $\mathbf{x} \in \{-1, 1\}^n$ is the total number of -1 's present in \mathbf{x} . This is the total number of 1's if we consider the string of 0's and 1's. The set $E_{n,k}$ denotes the set of all n -length points whose weight is k , i.e., $E_{n,k} = \{\mathbf{x} : \mathbf{x} \in \{-1, 1\}^n \text{ and } wt(\mathbf{x}) = k\}$. Here $|E_{n,k}| = \binom{n}{k}$. It can be noticed that $E_{n,k}$ is a restricted domain, where the restriction is that the all the points in $E_{n,k}$ will be of length n and weight k .

Autocorrelation of an n -variable Boolean function $f : \{0, 1\}^n \rightarrow \{0, 1\}$ is defined by

$$\mathcal{A}_f(\mathbf{a}) = \sum_{\mathbf{x} \in \{0, 1\}^n} (-1)^{f(\mathbf{x}) \oplus f(\mathbf{x} \oplus \mathbf{a})}, \mathbf{a} \in \{0, 1\}^n$$

It can be noticed this definition of autocorrelation can not be used directly to compute autocorrelation of f in $E_{n,k}$. As if we take any $\mathbf{x} \in E_{n,k}$ and take any $\mathbf{a} \in \{0, 1\}^n$ then $\mathbf{x} \oplus \mathbf{a}$ may not belong to $E_{n,k}$. For an $\mathbf{x} \in E_{n,k}$, we need to select an \mathbf{a} selectively such that $\mathbf{x} \oplus \mathbf{a}$ should also belong to $E_{n,k}$. As we have already pointed out (see the discussion in Section 3.2 below), significant bias could be identified in $\mathcal{A}_f(\mathbf{a})$ when $wt(\mathbf{a}) = 1$. In a similar line, we consider a special case, where a specific input bit will be selected, where the differential will exist. However, the weight of the two inputs should be of the same weight.

Let f be an n -variable Boolean function. Let S_1 and S_2 be two sets defined as $S_1 = \{\mathbf{x} \in E_{n,k} \mid u\text{-th bit of } \mathbf{x} \text{ is } -1\}$, $S_2 = \{\mathbf{x} \in E_{n,k} \mid u\text{-th bit of } \mathbf{x} \text{ is } 1\}$. Note

that $E_{n,k} = S_1 \cup S_2$ and $S_1 \cap S_2 = \emptyset$. The restricted autocorrelation of f over $E_{n,k}$ is defined as

$$\mathcal{A}_f^{E_{n,k}} = \sum_{\mathbf{x}_1 \in S_1, \mathbf{x}_2 \in S_2} (-1)^{f(\mathbf{x}_1) \oplus f(\mathbf{x}_2)}.$$

It is evident that $|S_1| = \binom{n-1}{k-1}$ and $|S_2| = \binom{n-1}{k}$. We are not concerned about the bit position u as it will be proved that this expression actually does not depend on u for an n -length Arbiter PUF.

The purpose of defining restricted autocorrelation is to study the autocorrelation spectrum of PUF in a restricted domain, where the simple construction of Arbiter PUF does not provide any bias.

2.2 Arbiter PUF as a Boolean function

Arbiter-based Physical Unclonable Functions (Arbiter PUFs) take an n -bit challenge and, based on the manufacturing process variation, provide a pseudorandom output of either 1 or 0. Here, we try to give a model of a PUF circuit using an additive delay model. We assume the total path delay, denoted by $\Delta(n)$, through the entire circuit is the sum of the path delays of elementary components. If an adversary knows all the elementary component delays, then response prediction for any challenge by calculating the circuit delay will become very easy.

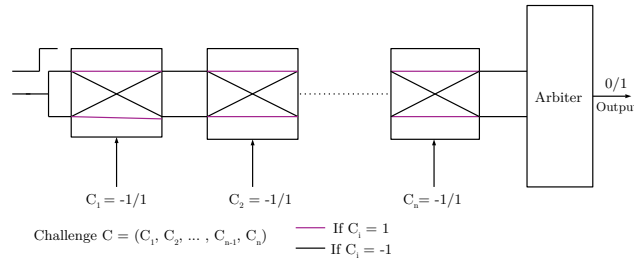


Figure 2-1: Basic structure of an Arbiter based PUF

An Arbiter PUF can be described as a function of the challenge and the path delays resulting from the n arbiter switches. Now, from Fig.2-1, we can say for n many switches we have 2^n many possible paths. Let $\delta_{top}(n)$ be the signal delay of the top path from the starting point to the end point at the n -th stage, and similarly $\delta_{bottom}(n)$ be the delay of the bottom path. For the sake of easier notation,

we use the transformation $a \rightarrow (-1)^a$, we map the challenge and response bits from $\{0, 1\} \rightarrow \{1, -1\}$ and from here we will assume $C_i \in \{-1, 1\}$ is the challenge bit at stage i . Let us introduce four fixed, different delay values at each switch i . The delay

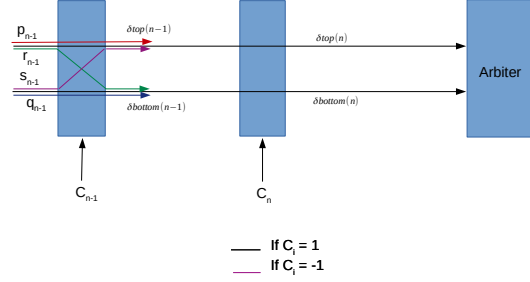


Figure 2-2: Dealy segment notation in each stage

segment notations p_n, q_n, r_n and s_n are shown in Fig.2-2. These p_n, q_n, r_n and s_n are randomly chosen from the same normal distribution. So if we know the delay values $\delta_{top}(i)$ and $\delta_{bottom}(i)$ at stage i , then we can derive $\delta_{top}(i+1)$ and $\delta_{bottom}(i+1)$ as a function of $\delta_{top}(i)$ and $\delta_{bottom}(i+1)$. At stage i , the delay segments are as follows:

$$\delta_{top}(i+1) = \frac{1}{2}(1 + C_{i+1})(p_{i+1} + \delta_{top}(i)) + \frac{1}{2}(1 - C_{i+1})(s_{i+1} + \delta_{bottom}(i))$$

$$\delta_{bottom}(i+1) = \frac{1}{2}(1 + C_{i+1})(q_{i+1} + \delta_{bottom}(i)) + \frac{1}{2}(1 - C_{i+1})(r_{i+1} + \delta_{top}(i))$$

where $C_i \in \{-1, 1\}$ is the challenge bit at the i^{th} stage.

Let $\Delta(i+1)$ be the difference between the top path delay $\delta_{top}(i+1)$ and bottom path delay $\delta_{bottom}(i+1)$ at stage $(i+1)$. Then,

$$\Delta(i+1) = C_{i+1} \cdot \Delta(i) + \alpha_{i+1} \cdot C_{i+1} + \beta_{i+1}$$

where,

$$\alpha_n = \frac{1}{2}(p_n - q_n + r_n - s_n)$$

$$\beta_n = \frac{1}{2}(p_n - q_n - r_n + s_n)$$

To simplify the above expression we define the parity challeng bits $P_k = \prod_{i=k+1}^n C_i$, where $P_n = 1$.

From above equation we can represent $\Delta(n)$ as a function of α_i, β_i and C_i for $1 \leq i \leq n$.

$$\Delta(0) = 0$$

$$\Delta(1) = C_1 \cdot \Delta(0) + \alpha_1 C_1 + \beta_1 = \alpha_1 C_1 + \beta_1$$

$$\Delta(2) = C_2(\alpha_1 C_1 + \beta_1) + \alpha_2 C_2 + \beta_2 = \alpha_1 C_1 C_2 + \beta_1 C_2 + \alpha_2 C_2 + \beta_2$$

.

.

.

$$\Delta(n) = C_n \cdot \Delta(n-1) + \alpha_n C_n + \beta_n$$

simplifying $\Delta(n)$ using $P_k = \prod_{i=k+1}^n C_i$ we get,

$$\Delta(n) = \alpha_1 P_0 + (\alpha_2 + \beta_1) P_1 + \dots + (\alpha_n + \beta_{n-1}) P_{n-1} + \beta_n P_n$$

Here $P_i \in \{-1, 1\}$, for $i = 0, 1, \dots, n$. Thus an n -length Arbiter PUF can be expressed as a function of delay parameters α_i, β_i and input $C = (C_1, C_2, \dots, C_n)$ as follows,

$$\Delta(C) = \alpha_1 P_0 + (\alpha_2 + \beta_1) P_1 + \dots + (\alpha_n + \beta_{n-1}) P_{n-1} + \beta_n P_n \quad (2.1)$$

Now, if the sign of $\Delta(C)$ is positive, then the PUF outputs 1 and if the sign of $\Delta(C)$ is negative, then the PUF outputs 0. From Equation 2.1, we can say that an n -length Arbiter PUF can be seen as a Boolean function which takes n -bit input $C = (C_1, C_2, \dots, C_n)$ and based on the sign of $\Delta(C)$, it outputs either 1 or 0.

For a fixed value of $C \in \{-1, 1\}^n$, the mathematical model of PUF (in Equation (2.1)) can also be re-written in a compact form $M_j \cdot X$, where $M_j = [P_0, \dots, P_{n-1}, 1]$ and $X = [\alpha_1, \alpha_2 + \beta_1, \dots, \alpha_n + \beta_{n-1}, \beta_n]^T$. That means the form of the complete equations corresponding to all values of $C \in \{-1, 1\}^n$ can be expressed as $M \cdot X$, where $M = [M_1, \dots, M_{2^n}]^T$ and $X = [\alpha_1, \alpha_2 + \beta_1, \dots, \alpha_n + \beta_{n-1}, \beta_n]^T$.

2.3 Modeling XOR of Arbiter-based PUFs as Boolean functions

A k -XOR Arbiter PUF consists of k parallel Arbiter PUFs whose outputs are XOR-ed to produce the pseudorandom bit. In this thesis, we consider the $k = 2$ case and refer

to the 2-XOR Arbiter PUF as an XOR-PUF.

Throughout the thesis, we consider two different classes of XOR-PUFs: the first one, where the lengths of the individual PUFs are the same, say n , and the second one, where the lengths of the individual PUFs are non-equal, say $n \neq m$.

2.3.1 XOR-PUF from the same length Arbiter PUFs

Here we feed an n -bit input to two n -length Arbiter PUFs with delay parameters α_i, β_i and α'_i, β'_i , respectively, where

$$\alpha_i = \frac{p_i - q_i}{2} + \frac{r_i - s_i}{2}, \beta_i = \frac{p_i - q_i}{2} - \frac{r_i - s_i}{2}, 1 \leq i \leq n,$$

$$\alpha'_i = \frac{p'_i - q'_i}{2} + \frac{r'_i - s'_i}{2}, \beta'_i = \frac{p'_i - q'_i}{2} - \frac{r'_i - s'_i}{2}, 1 \leq i \leq n,$$

and $P_j = \prod_{k=j+1}^n C_k$, $0 \leq j \leq n - 1$. Here, the output is generated by XOR-ing the output of the individual Arbiter PUFs as in Figure 2-3 and the resultant function becomes an n -variable Boolean function. Though, not necessarily a PUF, by abuse of notation, we shall call the mathematical model of the XOR of two PUFs, also a delay difference $\Delta^{XOR}(C)$, and, as described in [RSS⁺10b], be defined as follows:

$$\Delta^{XOR}(C) = \Delta(C) \cdot \Delta'(C). \quad (2.2)$$

We are actually interested in the sign of the above expression. If for a challenge $C \in \{-1, 1\}^n$, the sign of $\Delta^{XOR}(C)$ is positive, then the output from the XOR-PUF will be 1, and if the sign of $\Delta^{XOR}(C)$ is negative, then the output will be 0. In this thesis, we will be using the notation $\mathcal{B}_n^{XOR-PUF}$ to denote the set of n -variable Boolean functions exhaustively generated by the XOR of two n -length Arbiter PUFs.

2.3.2 XOR-PUF from different lengths Arbiter PUFs

Here we feed two different inputs of size n and m to two different Arbiter PUFs of lengths n and m , respectively. The corresponding delay parameters are given by α_i, β_i and α'_i, β'_i , respectively. Here, the output is generated by XOR-ing the outputs from the individual Arbiter PUFs as in Figure 2-4 and the resultant function becomes an $(n + m)$ -variable Boolean function. Let $C_1 \in \{-1, 1\}^n$ and $C_2 \in \{-1, 1\}^m$ be the

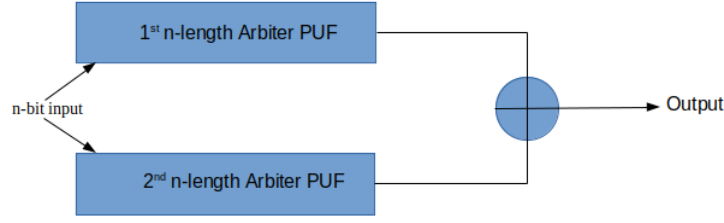


Figure 2-3: XOR-PUF from same length Arbiter PUFs having same inputs

inputs of two individual Arbiter PUFs of length n and m respectively. Then the delay difference $\Delta_{n,m}^{XOR}(C)$ (though, the mathematical model is not necessarily of a PUF, by abuse, we shall still call it delay difference) is given by the following equation

$$\Delta_{n,m}^{XOR}(C) = \Delta(C_1) \cdot \Delta'(C_2), \quad (2.3)$$

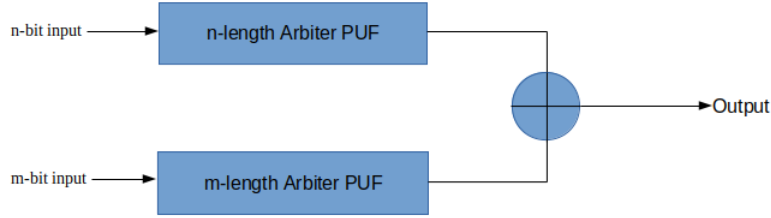


Figure 2-4: XOR-PUF from different length Arbiter PUFs

where $C = C_1 \parallel C_2$ is the input of length $(n + m)$. Here the output also depends on the sign of $\Delta_{n,m}^{XOR}(C)$. For a challenge $C \in \{-1, 1\}^{n+m}$, if the sign of $\Delta_{n,m}^{XOR}(C)$ is positive then the output will be 1 otherwise 0. We will be using the notation $\mathcal{B}_{n,m}^{XOR-PUF}$ to denote the set of $(n + m)$ -variable Boolean functions exhaustively generated from the XOR of two Arbiter PUFs of length n and m , respectively.

2.4 Modeling Priority Arbiter-based PUF as Boolean functions

Like Arbiter PUF, a Priority Arbiter-based PUF (PA-PUF) also can be expressed in terms of the challenge inputs and the delays of the paths. In this section we develop

the mathematical model of PA-PUF with three paths namely top (T), center (C) and bottom (B). Given a common impulse as input at the first stage the impulse reaches to the final stage via travelling through all stages. The challenge input $c[i]$ to stage i is either 1 or -1 . Depending upon the value of challenge input $c[i]$ the path alteration of the impulse occurs at the i -th stage of PA-PUF. The path alteration of the impulse at i -th stage occurs according to the following rules.

- $c[i] = 1$: $C \rightarrow T, B \rightarrow C, T \rightarrow B$.
- $c[i] = -1$: $B \rightarrow T, T \rightarrow C, C \rightarrow B$.

Depending upon the several device specific and environmental parameters of the PA-PUF some amount of delay occurs in each path during the transmission of the impulse. Due to this we associate different delay parameters with each of the three paths of a stage. For developing the mathematical modeling the delay parameters are taken randomly from a normal distribution with mean μ and standard deviation σ i.e., $\mathcal{N}(\mu, \sigma)$. We will use the following notations for our discussion.

- $\delta_T(i)$: delay in the top path at i -th stage.
- $\delta_C(i)$: delay in the center path at i -th stage.
- $\delta_B(i)$: delay in the bottom path at i -th stage.
- $\Delta_{CB}(i) = \delta_C(i) - \delta_B(i)$, delay difference between center path and bottom path.
- $\Delta_{TC}(i) = \delta_T(i) - \delta_C(i)$, delay difference between top path and center path.
- $\Delta_{BT}(i) = \delta_B(i) - \delta_T(i)$, delay difference between bottom path and top path.

From Figure 2-5 one can get an idea about the delays of different paths at the i -th stage after the alteration of the paths of the impulse.

We first measure the delay of the impulse in each of the three paths. This will help us to find an iterative formula for determining the delay of the impulse in each path at i -th stage from the delays of the impulse at $(i-1)$ -th stage. For the challenge input $c[i] = 1$, impulse in the center path (C) diverts to top path T with a delay $q_i^{(T)}$ and for $c[i] = -1$, the impulse in the bottom path (B) diverts to top path T with a delay $r_i^{(T)}$. Thus the delay at top path at i -th stage is,

$$\delta_T(i) = \frac{1 + c[i]}{2} [\delta_C(i-1) + q_i^{(T)}] + \frac{1 - c[i]}{2} [\delta_B(i-1) + r_i^{(T)}].$$

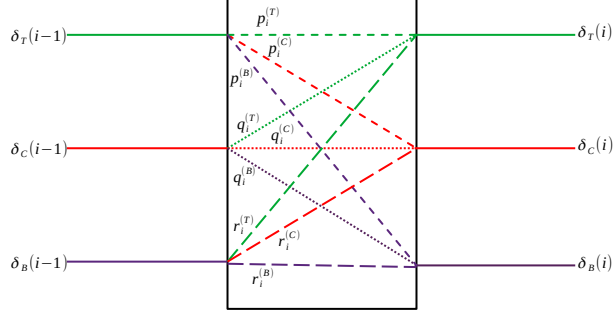


Figure 2-5: i -th stage of PA-PUF with three paths

For $c[i] = 1$, the impulse in the bottom path (B) diverts to center path (C) with a delay $r_i^{(C)}$ and for $c[i] = -1$, the impulse in the top path (T) diverts to center path C with delay $p_i^{(C)}$. Thus the delay at center path at i -th stage is,

$$\delta_C(i) = \frac{1 + c[i]}{2} [\delta_B(i-1) + r_i^{(C)}] + \frac{1 - c[i]}{2} [\delta_T(i-1) + p_i^{(C)}].$$

Similarly the delay in bottom path at i -th stage will be,

$$\delta_B(i) = \frac{1 + c[i]}{2} [\delta_T(i-1) + p_i^{(B)}] + \frac{1 - c[i]}{2} [\delta_C(i-1) + q_i^{(B)}].$$

Now we compute $\Delta_{CB}(i)$, $\Delta_{TC}(i)$ and $\Delta_{BT}(i)$.

$$\begin{aligned} \Delta_{CB}(i) &= \delta_C(i) - \delta_B(i) \\ &= \frac{1 + c[i]}{2} [\delta_B(i-1) + r_i^{(C)}] + \frac{1 - c[i]}{2} [\delta_T(i-1) + p_i^{(C)}] \\ &\quad - \left\{ \frac{1 + c[i]}{2} [\delta_T(i-1) + p_i^{(B)}] + \frac{1 - c[i]}{2} [\delta_C(i-1) + q_i^{(B)}] \right\} \\ &= \frac{1 + c[i]}{2} \left\{ \delta_B(i-1) + r_i^{(C)} - \delta_T(i-1) - p_i^{(B)} \right\} \\ &\quad + \frac{1 - c[i]}{2} \left\{ \delta_T(i-1) + p_i^{(C)} - \delta_C(i-1) - q_i^{(B)} \right\} \\ &= \frac{1 + c[i]}{2} \left\{ \Delta_{BT}(i-1) + r_i^{(C)} - p_i^{(B)} \right\} \\ &\quad + \frac{1 - c[i]}{2} \left\{ \Delta_{TC}(i-1) + p_i^{(C)} - q_i^{(B)} \right\} \\ &= \frac{1}{2} \left(\Delta_{BT}(i-1) + \Delta_{TC}(i-1) \right) + \frac{c[i]}{2} \left(\Delta_{BT}(i-1) - \Delta_{TC}(i-1) \right) \\ &\quad + \frac{c[i]}{2} (r_i^{(C)} - p_i^{(B)} - p_i^{(C)} + q_i^{(B)}) \end{aligned} \tag{2.4}$$

$$+\frac{1}{2}(r_i^{(C)} - p_i^{(B)} + p_i^{(C)} - q_i^{(B)})$$

$$\begin{aligned}
\Delta_{TC}(i) &= \delta_T(i) - \delta_C(i) \\
&= \frac{1+c[i]}{2} [\delta_C(i-1) + q_i^{(T)}] + \frac{1-c[i]}{2} [\delta_B(i-1) + r_i^{(T)}] \\
&\quad - \left\{ \frac{1+c[i]}{2} [\delta_B(i-1) + r_i^{(C)}] + \frac{1-c[i]}{2} [\delta_T(i-1) + p_i^{(C)}] \right\} \\
&= \frac{1+c[i]}{2} \left\{ \delta_C(i-1) + q_i^{(T)} - \delta_B(i-1) - r_i^{(C)} \right\} \\
&\quad + \frac{1-c[i]}{2} \left\{ \delta_B(i-1) + r_i^{(T)} - \delta_T(i-1) - p_i^{(C)} \right\} \\
&= \frac{1+c[i]}{2} \left\{ \Delta_{CB}(i-1) + q_i^{(T)} - r_i^{(C)} \right\} \\
&\quad + \frac{1-c[i]}{2} \left\{ \Delta_{BT}(i-1) + r_i^{(T)} - p_i^{(C)} \right\} \\
&= \frac{1}{2} \left(\Delta_{CB}(i-1) + \Delta_{BT}(i-1) \right) + \frac{c[i]}{2} \left(\Delta_{CB}(i-1) - \Delta_{BT}(i-1) \right) \\
&\quad + \frac{c[i]}{2} \left(q_i^{(T)} - r_i^{(C)} - r_i^{(T)} + p_i^{(C)} \right) \tag{2.5} \\
&\quad + \frac{1}{2} \left(q_i^{(T)} - r_i^{(C)} + r_i^{(T)} - p_i^{(C)} \right)
\end{aligned}$$

$$\begin{aligned}
\Delta_{BT}(i) &= \delta_B(i) - \delta_T(i) \\
&= \frac{1+c[i]}{2} [\delta_T(i-1) + p_i^{(B)}] + \frac{1-c[i]}{2} [\delta_C(i-1) + q_i^{(B)}] \\
&\quad - \left\{ \frac{1+c[i]}{2} [\delta_C(i-1) + q_i^{(T)}] + \frac{1-c[i]}{2} [\delta_B(i-1) + r_i^{(T)}] \right\} \\
&= \frac{1+c[i]}{2} \left\{ \delta_T(i-1) + p_i^{(B)} - \delta_C(i-1) - q_i^{(T)} \right\} \\
&\quad + \frac{1-c[i]}{2} \left\{ \delta_C(i-1) + q_i^{(B)} - \delta_B(i-1) - r_i^{(T)} \right\} \\
&= \frac{1+c[i]}{2} \left(\Delta_{TC}(i-1) + p_i^{(B)} - q_i^{(T)} \right) \\
&\quad + \frac{1-c[i]}{2} \left(\Delta_{CB}(i-1) + q_i^{(B)} - r_i^{(T)} \right) \\
&= \frac{1}{2} \left(\Delta_{TC}(i-1) + \Delta_{CB}(i-1) \right) + \frac{c[i]}{2} \left(\Delta_{TC}(i-1) - \Delta_{CB}(i-1) \right) \\
&\quad + \frac{c[i]}{2} \left(p_i^{(B)} - q_i^{(T)} - q_i^{(B)} + r_i^{(T)} \right) \tag{2.6}
\end{aligned}$$

$$+\frac{1}{2}\left(p_i^{(B)} - q_i^{(T)} + q_i^{(B)} - r_i^{(T)}\right)$$

Equations (2.4),(2.5),(2.6) are the iterative formulae for measuring the delay differences of the impulse between three paths for $i \geq 0$. Here we impose an initial condition $\Delta_{CB}(-1) = \Delta_{TC}(-1) = \Delta_{BT}(-1) = 0$. The summary of this result is presented in a form of a theorem in Theorem 1.

Theorem 1. *The delay differences of the impulse between three paths at the i -th stage of a PA-PUF are as per the Equations (2.4),(2.5),(2.6), where $\Delta_{CB}(-1) = \Delta_{TC}(-1) = \Delta_{BT}(-1) = 0$; $p_i^{(j)}, q_i^{(j)}, r_i^{(j)}$ follow $\mathcal{N}(\mu, \sigma)$ for $j \in \{T, C, B\}$ and $c[i] \in \{-1, 1\}$ is the challenge input to the i -th stage.*

We have observed that the delay differences $\Delta_{TC}(n), \Delta_{CB}(n), \Delta_{BT}(n)$ at any stage $n \geq 0$ satisfy the relation $\Delta_{TC}(n) + \Delta_{CB}(n) + \Delta_{BT}(n) = 0$. The supporting proof is given in the following proposition.

Proposition 1. $\Delta_{TC}(n) + \Delta_{CB}(n) + \Delta_{BT}(n) = 0, \forall n \geq 0$.

Proof. We will use mathematical induction to prove this result. From the above expressions we will get $\Delta_{TC}(0) + \Delta_{CB}(0) + \Delta_{BT}(0) = 0$. Let us assume for $i \geq 0, \Delta_{TC}(i) + \Delta_{CB}(i) + \Delta_{BT}(i) = 0$. Now for $n = i + 1$,

$$\begin{aligned} \Delta_{TC}(i+1) + \Delta_{CB}(i+1) &= \Delta_{BT}(i) + \frac{1}{2} \cdot (\Delta_{TC}(i) + \Delta_{CB}(i)) \\ &+ \frac{c[i]}{2} \cdot (\Delta_{CB}(i) - \Delta_{TC}(i)) + \frac{c[i]}{2} \cdot \left(-p_i^{(B)} + q_i^{(T)} + q_i^{(B)} - r_i^{(T)}\right) \\ &+ \frac{1}{2} \cdot \left(-p_i^{(B)} + q_i^{(T)} - q_i^{(B)} + r_i^{(T)}\right) \\ &= \Delta_{BT}(i) + \Delta_{TC}(i) + \Delta_{CB}(i) - \Delta_{BT}(i+1) \\ &= -\Delta_{BT}(i+1) \end{aligned}$$

Hence $\forall n \geq 0, \Delta_{TC}(n) + \Delta_{CB}(n) + \Delta_{BT}(n) = 0$. □

The delay differences are measured by an Arbiter which is placed after the n -th stage. Depending upon the sign of $\Delta_{TC}(n), \Delta_{CB}(n)$ and $\Delta_{BT}(n)$ it produces either 0 or 1 as an output. We associate the following six conditions on $\Delta_{TC}(n), \Delta_{CB}(n)$ and $\Delta_{BT}(n)$ for each possible outputs from the PA-PUF as defined in [SBP+22].

1. if $\Delta_{TC}(n) > 0$ and $\Delta_{CB}(n) > 0$ output is 1.

2. if $\Delta_{BT}(n) < 0$ and $\Delta_{CB}(n) < 0$ output is 1.
3. if $\Delta_{BT}(n) > 0$ and $\Delta_{TC}(n) > 0$ output is 0.
4. if $\Delta_{CB}(n) < 0$ and $\Delta_{TC}(n) < 0$ output is 0.
5. if $\Delta_{CB}(n) > 0$ and $\Delta_{BT}(n) > 0$ output is 0.
6. if $\Delta_{TC}(n) < 0$ and $\Delta_{BT}(n) < 0$ output is 1.

It can be noticed that an n -length PA-PUF outputs 1 if and only if any of the following relation holds,

- $\delta_T(n-1) > \delta_C(n-1) > \delta_B(n-1)$.
- $\delta_C(n-1) < \delta_B(n-1) < \delta_T(n-1)$.
- $\delta_B(n-1) < \delta_T(n-1) < \delta_C(n-1)$.

Equivalently we can say that an n -length PA-PUF outputs 1 if and only if $\delta_B(n-1) < \delta_T(n-1)$. Thus the output of an n -length PA-PUF can be characterized by the sign of $\Delta_{BT}(n-1)$ only. An n -length PA-PUF outputs 1 if and only if $\Delta_{BT}(n-1) < 0$ and outputs 0 if and only if $\Delta_{BT}(n-1) > 0$.

After a detailed discussion related to PUFs, now we will shift to other applications where Boolean functions are exploited in a different manner.

2.5 Evaluation of Randomness through Boolean functions

Random number generators find extensive utility in communication and cryptography. Generally speaking, the security of a protocol depends on the ‘randomness’ imparted by pseudo-random number generators. Therefore, it is crucial to scrutinize whether a stream that appears random exhibits any underlying non-random patterns. In this context, our focus lies on the *BoolTest* [SKŠ17], wherein each block of the bit-stream undergoes evaluation by a carefully selected Boolean function, and the resulting output bits are analyzed.

Consider N -bit of data \mathcal{D} , whose randomness we would like to test. The data is divided into non-overlapping blocks of m bits. For simplicity, we consider N is

divisible by m , i.e., there are $n = \frac{N}{m}$ blocks. Each block of data is applied to a Boolean function of m -input bits to obtain one-bit output. Thus, given N -bit data, we obtain $\frac{N}{m}$ bits out of the Boolean function. Let us call the collection of all m block inputs obtained in such a manner \mathcal{I} . The collection \mathcal{I} is a multi-set, not a set since there might be m length blocks that occur more than once.

Naturally, we need to study the frequency distribution from the set \mathcal{I} to identify any non-randomness. By the method described in *BoolTest* [SKŠ17], one may try to find the best distinguisher function on the frequency distribution obtained. The method used in *BoolTest* to obtain the Boolean function for the best distinguisher involves the metric Z -score. Before defining Z -score in our interpretation, let us introduce some notations. Support of an m -input 1-output Boolean function f is defined as

$$\text{supp}(f) = \{x \in \{0, 1\}^m : f(x) = 1\}. \quad (2.7)$$

We also define $W(f)$ as the following set,

$$W(f) = \{x \in \mathcal{I} : f(x) = 1\}. \quad (2.8)$$

Let q_f be the proportion of inputs for which the f returns 1, i.e.,

$$q_f = \frac{|\text{supp}(f)|}{2^m}. \quad (2.9)$$

If the input distribution had uniformly been random, we would get each input $x \in \{0, 1\}^m$ with equal probability. If we had a uniform distribution of m -block inputs $\{0, 1\}^m$, then the number of data blocks for which the function f will output 1 is nq_f . Let p_f be the proportion of elements in \mathcal{I} which will output 1 when given as an input to f .

$$p_f = \frac{|\{x \in \mathcal{I} : f(x) = 1\}|}{2^m} = \frac{|W(f)|}{2^m}. \quad (2.10)$$

In other words, the number of inputs for which the function f will output 1 from the collection \mathcal{I} will be np_f .

Definition 1 (Z -score). *For a given function f of m -variables, and a collection \mathcal{I} ,*

p_f and q_f as defined above, the Z -score defined in [SKŠ17] is given as,

$$z_f = \frac{\#1 - nq_f}{\sqrt{nq_f(1 - q_f)}}. \quad (2.11)$$

where $\#1$ is the random variable that describes the number of m -bit blocks in the input data \mathcal{D} , that when fed to the function f returns 1. Here, $\#1$ can be written as

$$\#1 = |W(f)| = np_f \quad (2.12)$$

Thus the Z -score would be

$$z_f = \left| \frac{np_f - nq_f}{\sqrt{nq_f(1 - q_f)}} \right|. \quad (2.13)$$

To provide more intuition to the definition above, if Y is a random variable representing the number of ones obtained as output from the function f over some input distribution, then Z -score is the normalization of binomially distributed random variable Y .

Now that we have defined Z -score, let us proceed to outline a brief introduction to the method presented by [SKŠ17]. Note that, we are interested in identifying the most optimal Boolean function. Thus, the Z -score we discuss will always be related to a suitable Boolean function f , and thus, as in Definition 1, we always have z_f that is to be studied. Note that the number of distinct m -variable Boolean functions is 2^{2^m} and choosing the optimal Boolean function out of that super-exponential space is the main challenge.

2.5.1 Brief Description of *BoolTest* by Šýs et al [SKŠ17]

The basic idea is to construct an m -bit Boolean function that will produce the highest Z -score. Given that there are 2^{2^m} Boolean functions, it has been commented in [SKŠ17] that only a heuristic method in the set of m -variable Boolean functions will be attempted to identify the function. We will later show that this technique [SKŠ17] is sub-optimal and consequently we will present an optimal algorithm. Now let us explain the strategy of [SKŠ17].

The *BoolTest* algorithm $\mathcal{B}(deg, m, t, k)$ takes in as input the following parameters:

- deg : Each term in the ANF of the functions searched by *BoolTest* would be of degree deg .
- m : Block size, which is also the number of inputs to the Boolean functions.
- t : Top t monomials of degree deg are chosen and combined by XOR in the next step.
- k : Distinguishers are formed by combining k many monomials of degree deg .

Algorithm 1: BoolTest $\mathcal{B}(deg, m, t, k)$

```

1  $M \leftarrow \{1, \dots, m\}$ ;
2  $T \leftarrow \{1, \dots, t\}$ ;
   // GET-SUBSETS( $j, M$ ) returns all subsets of  $M$  of size  $j$ 
3  $S_{deg} \leftarrow \text{GET-SUBSETS}(deg, M)$ ;
   // Set  $F_{deg}$  contains all monomials of degree  $deg$ 
4  $F_{deg} \leftarrow \{f : f = \prod_{j \in J} x_j, \forall J \in S_{deg}\}$ ;
   // Choose top  $t$  functions from  $F_{deg}$  with highest  $Z$ -score
   // GET-MAX function takes  $t$  monomials with highest  $Z$ -score
5  $F_t \leftarrow \text{GET-MAX}(t, F_{deg})$ ;
6  $S_k \leftarrow \text{GET-SUBSETS}(k, T)$ ;
   // Take combinations of  $k$  monomials from  $F_t$ 
7  $F = \{f : f = \bigoplus_{k \in K} f_k, \forall K \in S_k, f_k \in F_t\}$ ;
   // Return the max  $Z$ -score and the corresponding distinguisher function from
    $F$ 
8 return  $\arg \max_{f \in F} z(f)$ ;
```

The Algorithm 1 given above provides an algorithmic outline of the *BoolTest* [SKŠ17]. There are many other details related to optimization to improve the performance of the algorithm. To understand the approach, let us consider an example. If the given parameters are $\mathcal{B}(deg = 2, m = 4, t = 5, k = 3)$, then *BoolTest* first computes Z -score for all the monomials of degree 2 of the form $x_i x_j$ where $i, j \in \{1, 2, 3, 4\}$ (since $m = 4$). There are six such degree 2 monomials in this case. From these monomials, the top $t = 5$ monomials with the highest Z -score will be selected. Now, choose $k = 3$ monomials out of the 5 obtained in the first step (total $\binom{5}{3}$ combinations) and combine them using XOR operation to form a new function with 3 monomials each

having degree 2, and obtain the Z -score for each combination. The function with the maximum Z -score out of these combinations is considered to be the function that will provide the best distinguisher through this heuristic.

Let the data \mathcal{D} be a sequence of n random variables X_1, \dots, X_n . The null hypothesis is,

$$H_0 : X_i \sim \text{Uniform}(0, 2^m - 1), \forall i \in \{1, \dots, n\} \quad (2.14)$$

If the Boolean function f had been fixed, the number of ones ($\#1$) would be a random variable that follows Binomial distribution $B(n, q_f)$ and z_f would approximately follow standard normal distribution $\mathcal{N}(0, 1)$.

The highest Z -score would be of the form $Z = \max\{z_{f_1}, \dots, z_{f_{2^m}}\}$ where each z_{f_i} approximately follows standard normal distribution and, its CDF would be difficult to calculate as the z_{f_i} 's are not independent.

The value [SKŠ17] calculates is the random variable $\max\{z_{f_1}, \dots, z_{f_{\binom{t}{k}}}\}$ (it considers only $\binom{t}{k}$ among all possible Boolean functions, $f_1, f_2, \dots, f_{\binom{t}{k}}$ are functions constructed from the data) whose CDF would be similarly difficult to calculate. Instead, [SKŠ17] estimates the acceptance region using a “reference window” created by running the process on (assumed) true random data. If the highest Z -score achieved by the procedure on some sample of data falls within the reference window, the data is assumed to be random; otherwise, it is considered that the data might have non-randomness. However, in Chapter 6, we observe that very high Z -scores do not necessarily indicate non-randomness. For example, for large block size m , and the amount of data much smaller than 2^m blocks, very high Z -scores are possible even for truly random data.

Next we shift to our final contribution of the thesis where Boolean functions are applied in the study of two-party nonlocal games.

2.6 Two-Party Nonlocal Games in terms of Boolean functions

Nonlocal games refer to the games played between multiple space-separated players and a referee where communication between the players is strictly forbidden during the game. In a binary input binary output nonlocal game, the referee sends an input bit to each player, who then responds by sending output bits to the referee. In the classical scenario, players fix some strategies among themselves based on the winning condition prior to the game's commencement. Whereas, in the quantum scenario, players establish shared entanglement among themselves before the start of the game, aiming to gain an added advantage in the winning probability compared to the classical scenario.

Definition 2 (CHSH Game). *The most well known binary input binary output two-party nonlocal game is the CHSH game [CHSH69] where a referee provides two uniformly random bits x and y to each of the two players. After receiving the inputs, the two parties send their output bits a and b to the referee. The players win the game against the referee if $(x \wedge y) = (a \oplus b)$. Thus, the Boolean function representation of the CHSH game is given by $f(x, y, a, b) = (x \wedge y) \oplus (a \oplus b)$, and they win the game against the referee if and only if $f = 0$.*

In the classical scenario, where $x \wedge y = 0$ in three out of four cases, the maximum winning probability of the CHSH game is 0.75 when Alice and Bob pre-decide to provide identical outputs. However, in the quantum scenario, the maximum success probability is $\cos^2 \pi/8$ (approximately 0.85) by sharing a maximally entangled state beforehand. The optimality of the quantum advantage has been proved by Cirel'son in [Cir80].

Based on the distribution of the successful outcomes (i.e., the distribution of 0's) in the output column of the Boolean function, a binary input binary output two-party nonlocal game can be represented in terms of *partitions* of the total number of successful outcomes.

Definition 3 (Partition of a nonlocal game). *A partition is a representation of a class of n party nonlocal games depending on the total number of successful outcomes. A partition of a nonlocal game is generated by splitting up the total number of successful*

outcomes into 2^n parts depending on the number of successful outcomes for each of the 2^n possible inputs. For an n -party binary input binary output nonlocal game with d number of successful outcomes (where $2^n \leq d \leq 2^{2^n}$), the corresponding partition will be represented as a summation of 2^n non-zero numbers (like $n_1 + n_2 + \dots + n_{2^n}$) such that $d = \sum_{i=1}^{2^n} n_i$ where each n_i is the number of successful outcomes for the i -th input such that $0 < n_i \leq 2^n$.

For a binary input binary output two-party nonlocal game, there are four possible inputs and for every input, there can have atmost four possible successful outcomes. So for these games, the partition representation is of the form $p_1 + p_2 + p_3 + p_4$ where each p_i denotes the total number of successful outcomes for the i -th input such that $0 \leq p_i \leq 4$. For example, one may consider the CHSH game (which represents a balanced 4-variable Boolean function) for which the partition representation is of the form $2 + 2 + 2 + 2$. Similarly, every other binary input binary output two-party nonlocal games can be represented as a summation of four non-zero numbers.

From these discussions, one can easily understand that many different games have the same representation of the partition. However, all the games that belong to a particular partition may not behave similarly. In this current endeavour, our focus lies in identifying all games that provide quantum advantages (i.e., a higher probability of winning in the quantum scenario compared to the classical scenario).

Definition 4 (Separation for a nonlocal game). *A separation denotes the difference between the maximum classical and the maximum quantum success probabilities for those games which offer a quantum advantage.*

For the sake of simplicity, from now onwards, we use the notation x and y to denote input bits and the notation a and b to denote the output bits of the two parties. $\bar{x}, \bar{y}, \bar{a}, \bar{b}$ denotes the usual complements (bit complement) of x, y, a, b respectively. Later on, if nothing is specified explicitly, whenever we use xy as input and ab as output for the two players, we assume that xy and ab can take any values from the set $\{00, 01, 10, 11\}$.

Definition 5 (GHZ Game). *Nonlocal games involving more than two parties, are referred to as pseudo-telepathy games [BBT05]. GHZ game is an example of a three-party nonlocal game that utilizes entanglement to demonstrate quantum supremacy [GHZ89]. In this game, each player (Alice, Bob, and Carol) receives a uniformly*

random bit x_1, x_2, x_3 from the referee, with possible inputs of $(0, 0, 0)$, $(1, 1, 0)$, $(1, 0, 1)$, and $(0, 1, 1)$. The players then send back a single bit output $a_1, a_2, a_3 \in \{0, 1\}$ to the referee. They win the game if and only if $(x_1 \vee x_2 \vee x_3) = (a_1 \oplus a_2 \oplus a_3)$.

In classical scenarios, where the players cannot utilize entanglement, the maximum winning probability of the GHZ game is 0.75. This can be achieved when the three players agree in advance to either all output 1 or only one of them outputs 1. However, in the quantum paradigm, by using the GHZ entangled state $\frac{1}{\sqrt{2}}(|000\rangle + |111\rangle)$, the players can win the game against the referee with a probability of 1, showing classical-quantum separation.

The GHZ game can also be represented using a (partial) Boolean function given by $f(x_1, x_2, x_3, a_1, a_2, a_3) = (x_1 \vee x_2 \vee x_3) \oplus (a_1 \oplus a_2 \oplus a_3)$, where the players win the game if $f = 0$, and the input bits (x_1, x_2, x_3) are restricted to the set $\{(0, 0, 0), (1, 1, 0), (1, 0, 1), (0, 1, 1)\}$.

How do the Arbiter PUFs Sample the Boolean Function Class?

Contents

3.1 Introduction	52
3.2 Motivation	52
3.3 Relation Between $\mathcal{B}_n^{\text{PUF}}$ and \mathcal{B}_n	54
3.4 Determining whether $f \in \mathcal{B}_n^{\text{PUF}}$	65
3.5 On Restricted Autocorrelation of Arbiter PUF	70
3.6 Conclusion	74

Arbiter-based PUF is a hardware based pseudorandom bit generator where the pseudorandomness in the output bits depends on device specific parameters. Our work shows how one can explore the functions achieved through an Arbiter PUF construction with random delay parameters. Our technique mostly shows limitation of such functions from the angle of cryptographic evaluation as the subclass of the Boolean function can be identified with much better efficiency (much less complexity) than random. In this direction we explain new ideas to distinguish the Arbiter PUFs from any other Boolean functions. On the other hand, we note that under certain constrains on the weights of inputs, such a simple model of Arbiter PUFs provides good cryptographic parameters in terms of differential analysis. In this regard, we theoretically solve the problem of autocorrelation properties in a restricted space of input variables with a fixed weight.

3.1 Introduction

As we have already described, an Arbiter-based PUF can act as an n -variables Boolean function from $\{0, 1\}^n$ to $\{0, 1\}$. In most of the cryptographic applications, the input bits of a Boolean function are usually considered independent and taken uniformly from the domain.

From [SBC⁺19b], it can be referred that if one generates output bits corresponding to two challenge inputs $C = (C_1, C_2, \dots, C_n)$ and $\tilde{C} = (\tilde{C}_1, \tilde{C}_2, \dots, \tilde{C}_n)$, where C and \tilde{C} belong to $\{-1, 1\}^n$ and differ only at the most significant bit (MSB) position (i.e., $C_1 + \tilde{C}_1 = 0$), then the output bits will match with high probability. The position of the differed challenge bit plays an important role in producing the bias. One can look into Figure 2-1 to understand the position of the challenge bits. This bias reduces with the location of the bit difference at the inputs. The least bias occurs for the middle-most bit. Naturally, this lack of randomness provides a direction that the PUF devices can only produce a restricted class of Boolean functions, not all. Consequently, the immediate scientific question is to explore the set of Boolean functions such Arbiter PUFs are generating. In this regard, here we present relevant combinatorial results to show certain necessary conditions regarding the existence or non-existence of Boolean functions generated out of the Arbiter PUFs. Then we try to find out for what kinds of combinatorial properties the functions from Arbiter PUFs resemble a randomly chosen Boolean function better. We also show how to distinguish the Arbiter PUFs from any other Boolean functions. We note that if one considers a certain autocorrelation measure after restricting the input bit pattern to a fixed weight, then such bias disappears. Thus, if one can restrict the attack model with such a constraint, then the use of Arbiter PUFs in certain applications (such as lightweight environment) might be recommended.

3.2 Motivation

Theoretical estimation of autocorrelation of an n -variable PUF over a complete domain $\{-1, 1\}^n$ is discussed in [SBC⁺19b]. In the same paper, it has also been shown that the outputs corresponding to inputs are heavily biased when two inputs differ at the first position. It means that the autocorrelation value of $f \in \mathcal{B}_n^{\text{PUF}}$ is not good for certain $\mathbf{a} \in \{0, 1\}^n$. To verify the theoretical analysis presented in [SBC⁺19b] we have

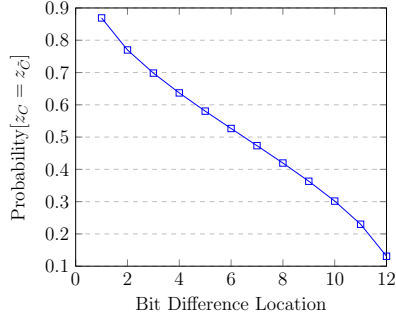


Figure 3-1: Representation of Table 3.1

Bit Difference Location	$\Pr[z_C = z_{\tilde{C}}]$
1	0.8691
2	0.7699
3	0.6982
4	0.6368
5	0.5804
6	0.5266
7	0.4734
8	0.4196
9	0.3632
10	0.3017
11	0.2300
12	0.1309

Table 3.1: Experimental Bias of PUFs ($n = 12$) in complete domain (over 1024 randomly chosen Arbiter PUFs) for single bit difference, matching with the theoretical values from [SBC⁺19b]

performed a simulation. We have taken random values of delay parameters from a normal distribution and generated 1024 many random 12-length PUFs. For each of these 12-length PUFs we have considered two inputs $C, \tilde{C} \in \{-1, 1\}^{12}$ where C, \tilde{C} differ only in one location i i.e., if $C = (C_1, \dots, C_i, \dots, C_{12})$ and $\tilde{C} = (C_1, \dots, -C_i, \dots, C_{12})$. For each of these two inputs we compute the output from the PUF, let z_C and $z_{\tilde{C}}$ be the respective output bits. Finally we compute $\Pr[z_C = z_{\tilde{C}}]$. The observed experimental observation is presented in Figure 3-1 and Table 3.1 and our observations are completely in the same direction of the theoretical result presented in [SBC⁺19b]. From Lemma 1 of [SBC⁺19b] it is also evident that the bias ϵ (i.e., $\Pr[z_C = z_{\tilde{C}}] = \frac{1}{2} \pm \epsilon$) increases with the increase of length of the PUF.

To understand the autocorrelation values we consider a 12-variable PUF and two inputs C and \tilde{C} where C and \tilde{C} differ at only one location. From the result of [SBC⁺19b] we know that the output z_C and $z_{\tilde{C}}$ are highly biased for certain bit difference locations. The experimental $\Pr[z_C = z_{\tilde{C}}]$ for different single bit difference locations is provided in Table 3.1. From Table 3.1 and Figure 3-1 it can be observed that the bias is highest when the bit difference location is either first or last and bias is least when the bit difference location is in the middle. Thus for certain values of $\mathbf{a} \in \{0, 1\}^n$ the expected autocorrelation value of $f \in \mathcal{B}_n^{\text{PUF}}$ significantly differs from

0.5.

To get a clearer idea about the autocorrelation distribution of PUF we perform statistical analysis. We consider all 4-variable Boolean functions and PUFs and measure the average number of Boolean functions and PUFs corresponding to different possible autocorrelation values $\{-16, -8, -4, 0, 4, 8, 12, 16\}$. From Figure 3-2 it can be observed that the distribution of PUF differs significantly from the distribution of Boolean function.

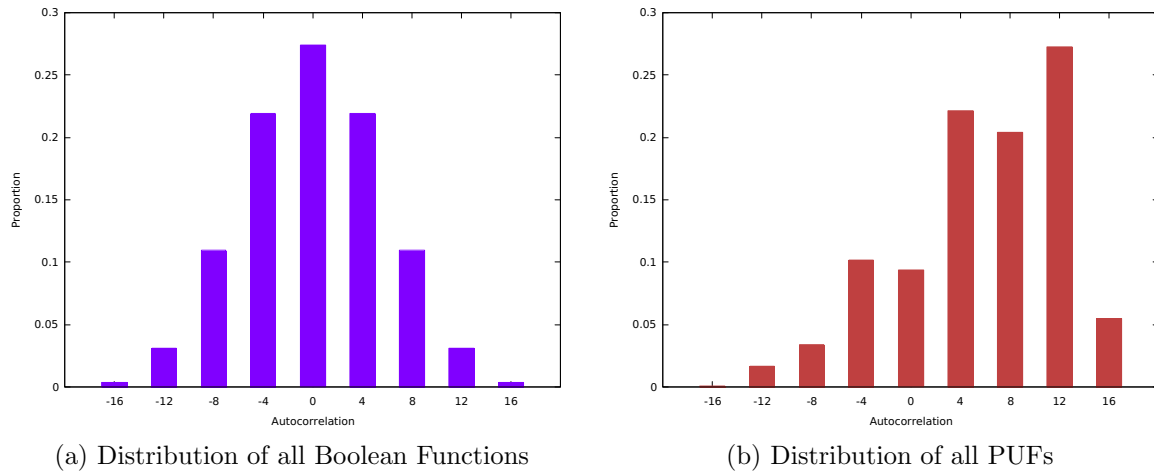


Figure 3-2: Comparison of Autocorrelation Distribution in Complete Domain

Now we provide a clear answer why the autocorrelation distribution is highly biased for PUF for single bit difference. The basic reason is that the Arbiter PUFs cannot exhaustively generate all possible Boolean functions. This observation motivates us to investigate the following.

- How to estimate the set $\mathcal{B}_n^{\text{PUF}}$?
- Can we obtain a restricted definition of autocorrelation so that the Arbiter PUFs do not expose a significant bias?

3.3 Relation Between $\mathcal{B}_n^{\text{PUF}}$ and \mathcal{B}_n

In this section, we explore the class of Boolean functions generated from n -variable PUFs i.e., $\mathcal{B}_n^{\text{PUF}}$. To compute the number of distinct Boolean functions which can be constructed using PUFs we start with $n = 1$. The total number of Boolean functions

involving 1-variable is $|\mathcal{B}_1| = 2^{2^1} = 4$. We all know that a PUF can be seen as a Boolean function. Thus, the obvious question is if we consider 1-length PUF, can that generate all possible Boolean functions given different delay parameters. To answer this question we state the following proposition.

Proposition 2. *All possible Boolean functions involving 1-variable can be generated by using 1-length PUFs i.e., $\mathcal{B}_1^{\text{PUF}} = \mathcal{B}_1$.*

Proof. This proposition can be proven by exhaustively enumerating $\mathcal{B}_1^{\text{PUF}}$. We have considered 1-length PUFs for different random delay parameters and observed that all the possible truth tables are generated in our experiment. Thus $|\mathcal{B}_1^{\text{PUF}}| = |\mathcal{B}_1| = 4$. \square

Now we move towards the case for $n = 2$. The total number of Boolean functions in this case is $|\mathcal{B}_2| = 2^{2^2} = 16$. Interestingly, from our experiments, we have observed that 14 many Boolean functions can be constructed from 2-length Arbiter PUFs, i.e., $|\mathcal{B}_2^{\text{PUF}}| = 14$. Truth tables of two specific Boolean functions can never be constructed using 2-length Arbiter PUFs. In this regard, we will state the following result.

Proposition 3. *The following two Boolean functions f_1 and f_2 do not belong to $\mathcal{B}_2^{\text{PUF}}$.*

C_2	C_1	f_1	f_2
1	1	1	0
1	-1	0	1
-1	1	1	0
-1	-1	0	1

Proof. The mathematical model of 2-length PUF is $\Delta(C) = \alpha_1 P_0 + (\alpha_2 + \beta_1) P_1 + \beta_2$, where $P_0 = C_1 C_2$ and $P_1 = C_2$. Here α_i, β_i are the delay parameters. We consider the truth table of f_1 first. It can be observed that if the $\text{sign}(\Delta(C))$ and $\text{sign}(C_1)$ are the same then only the truth table f_1 can be generated from 2-length PUF. Thus, to generate the same truth values from a 2-length PUF, we need to have the following scenarios.

C_2	C_1	$\Delta(C)$
1	1	$\alpha_1 + (\alpha_2 + \beta_1) + \beta_2 > 0$
1	-1	$-\alpha_1 + (\alpha_2 + \beta_1) + \beta_2 < 0$
-1	1	$-\alpha_1 - (\alpha_2 + \beta_1) + \beta_2 > 0$
-1	-1	$\alpha_1 - (\alpha_2 + \beta_1) + \beta_2 < 0$

If the above conditions hold for at least one pair of $\alpha_1, \alpha_2, \beta_1, \beta_2$ then only the truth values of f_1 can be generated. If we add two > 0 inequalities then we will have $\beta_2 > 0$ and if we add two < 0 inequalities then we will have $\beta_2 < 0$. This generates a contradiction. Hence the truth table of f_1 can not be generated from the 2-length Arbiter PUF structure. Similarly, it can be shown that it is not possible to generate the truth table of f_2 using a 2-length PUF. Thus $f_1, f_2 \notin \mathcal{B}_2^{\text{PUF}}$. \square

Using the transformation $a \rightarrow (-1)^a$ for $a \in \{0, 1\}$, the Algebraic Normal Form (ANF) of f_1, f_2 are $f_1(x_1, x_2) = 1 \oplus x_1$ and $f_2(x_1, x_2) = x_1$ respectively. Here x_1 corresponds to C_1 .

Proposition 3 justifies that $|\mathcal{B}_2^{\text{PUF}}| = 14$ as we noted from exhaustive experiment and directs us towards the following result.

Lemma 1. *For any n -variable Boolean function $f \notin \mathcal{B}_n^{\text{PUF}}$ if and only if $(1 \oplus f) \notin \mathcal{B}_n^{\text{PUF}}$.*

Proof. To prove this, we assume that there exists an n -variable Boolean function $f \in \mathcal{B}_n^{\text{PUF}}$ but $1 \oplus f \notin \mathcal{B}_n^{\text{PUF}}$. Let the n -length PUF be $\Delta(C) = \alpha_1 P_0 + (\alpha_2 + \beta_1) P_1 + (\alpha_3 + \beta_2) P_2 + \dots + (\alpha_n + \beta_{n-1}) P_{n-1} + \beta_n$. Here, α_i, β_i are the delay parameters. We know that depending on the sign of $\Delta(C)$, the truth table of $1 \oplus f$ is generated. Now if we consider a PUF with the delay parameters $\alpha'_i = -\alpha_i$ and $\beta'_i = -\beta_i$ and construct the PUF $\Delta(C)' = \alpha'_1 P_0 + (\alpha'_2 + \beta'_1) P_1 + (\alpha'_3 + \beta'_2) P_2 + \dots + (\alpha'_n + \beta'_{n-1}) P_{n-1} + \beta'_n$, then $\text{sign}(\Delta(C))$ and $\text{sign}(\Delta(C)')$ will be opposite for the same challenge values. Thus the truth table generated from $\Delta(C)'$ will be the truth table of $1 \oplus (1 \oplus f) = f$. Which contradicts our assumption. Hence if a Boolean function $f \notin \mathcal{B}_n^{\text{PUF}}$ then $(1 \oplus f) \notin \mathcal{B}_n^{\text{PUF}}$. Similarly if $(1 \oplus f) \notin \mathcal{B}_n^{\text{PUF}}$ then $f \notin \mathcal{B}_n^{\text{PUF}}$. \square

We know that any $(n + 1)$ -variable Boolean function f can be expressed as $f(x_1, \dots, x_{n+1}) = (1 \oplus x_{n+1})f_1(x_1, \dots, x_n) \oplus x_{n+1}f_2(x_1, \dots, x_n)$, where f_1, f_2 are two Boolean functions involving n variables. This is basically equivalent to $f(x_1, \dots, x_{n+1}) = f_1(x_1, \dots, x_n) \parallel f_2(x_1, \dots, x_n)$, in terms of concatenating the truth tables. That is, the truth table of f can be divided into two halves. In upper half if we consider $x_{n+1} = 0$, then it will contain the truth values of f_1 and in lower half if we consider $x_{n+1} = 1$ then it will contain the truth values of f_2 .

Every 3-variable Boolean function f can be written as $f = f_1 \parallel f_2$, where f_1 and f_2 are two Boolean functions involving 2 variables. As the constructions of PUFs

depend on parameters from normal distributions, the natural question is that if we consider a 3-variable PUF then can it be of the form $F = f \parallel f_1$ or $F = f_1 \parallel f$, where $f_1 = (1 \ 0 \ 1 \ 0) \notin \mathcal{B}_2^{\text{PUF}}$ (see Proposition 3) and $f \in \mathcal{B}_2$. The mathematical model of 3-variable PUF is $\Delta(C) = \alpha_1 P_0 + (\alpha_2 + \beta_1) P_1 + (\alpha_3 + \beta_2) P_2 + \beta_3$, where $P_0 = C_1 C_2 C_3$, $P_1 = C_2 C_3$ and $P_2 = C_3$. We prepare a truth table of a 3-variable PUF $F = f_1 \parallel f$ where $f_1 = (1 \ 0 \ 1 \ 0) \notin \mathcal{B}_2^{\text{PUF}}$ and $f \in \mathcal{B}_2$. We now break the truth table into two parts. In the upper part $C_3 = 1$ and in lower part $C_3 = -1$. Without loss of generality we consider $f = (0 \ 0 \ 0 \ 0)$. The final truth table of F will be of the following form.

C_3	C_2	C_1	$\Delta(C)$	$F = f_1 \parallel f$
1	1	1	$\alpha_1 + (\alpha_2 + \beta_1) + (\alpha_3 + \beta_2) + \beta_3 > 0$	1
1	1	-1	$-\alpha_1 + (\alpha_2 + \beta_1) + (\alpha_3 + \beta_2) + \beta_3 < 0$	0
1	-1	1	$-\alpha_1 - (\alpha_2 + \beta_1) + (\alpha_3 + \beta_2) + \beta_3 > 0$	1
1	-1	-1	$\alpha_1 - (\alpha_2 + \beta_1) + (\alpha_3 + \beta_2) + \beta_3 < 0$	0
-1	1	1	$-\alpha_1 - (\alpha_2 + \beta_1) - (\alpha_3 + \beta_2) + \beta_3 < 0$	0
-1	1	-1	$\alpha_1 - (\alpha_2 + \beta_1) - (\alpha_3 + \beta_2) + \beta_3 < 0$	0
-1	-1	1	$\alpha_1 + (\alpha_2 + \beta_1) - (\alpha_3 + \beta_2) + \beta_3 < 0$	0
-1	-1	-1	$-\alpha_1 + (\alpha_2 + \beta_1) - (\alpha_3 + \beta_2) + \beta_3 < 0$	0

We consider the following pairs of equations from the upper part of the above truth table.

$$\begin{cases} -\alpha_1 + (\alpha_2 + \beta_1) + (\alpha_3 + \beta_2) + \beta_3 < 0 \\ \alpha_1 - (\alpha_2 + \beta_1) + (\alpha_3 + \beta_2) + \beta_3 < 0 \end{cases} \quad (3.1)$$

$$\begin{cases} \alpha_1 + (\alpha_2 + \beta_1) + (\alpha_3 + \beta_2) + \beta_3 > 0 \\ -\alpha_1 - (\alpha_2 + \beta_1) + (\alpha_3 + \beta_2) + \beta_3 > 0 \end{cases} \quad (3.2)$$

From Equation (3.1) we get $(\alpha_3 + \beta_2) + \beta_3 < 0$ and from Equation (3.2) we get $(\alpha_3 + \beta_2) + \beta_3 > 0$, which is a contradiction. Thus for any $f \in \mathcal{B}_3^{\text{PUF}}$ it can not be of the form $f_1 \parallel f_2$ or $f_2 \parallel f_1$ where $f_1 = (1 \ 0 \ 1 \ 0) \notin \mathcal{B}_2^{\text{PUF}}$. Similarly we can prove that for any $f \in \mathcal{B}_3^{\text{PUF}}$ it can not be of the form $(1 \oplus f_1) \parallel f_2$ or $f_2 \parallel (1 \oplus f_1)$ where $f_1 \notin \mathcal{B}_2^{\text{PUF}}$. In this regard, we present the following important result.

Theorem 2. *If $f_1 \notin \mathcal{B}_n^{\text{PUF}}$, then there does not exist any $F \in \mathcal{B}_{n+1}^{\text{PUF}}$ of the form $f_1 \parallel f$ or $f \parallel f_1$.*

Proof. Assume that there exists an $F \in \mathcal{B}_{n+1}^{\text{PUF}}$ such that $F = f_1 \parallel f$ and $f_1 \notin \mathcal{B}_n^{\text{PUF}}$. Let the challenge input to the $(n+1)$ -variable PUF be $C = (C_1, \dots, C_{n+1})$. The mathematical model of the $(n+1)$ -variable PUF corresponding to F is

$$\Delta(C) = \alpha_0 P_0 + (\alpha_1 + \beta_0) P_1 + \dots + (\alpha_{n+1} + \beta_n) P_n + \beta_{n+1}, \quad (3.3)$$

where $P_k = \prod_{i=k+1}^{n+1} C_i$. As $F \in \mathcal{B}_{n+1}^{\text{PUF}}$, the inequalities constructed from $\Delta(C)$ in Equation (3.3) and the truth table corresponding to F will provide a solution for α_i and β_i . Let us look at the truth table of F into two equal parts. In the upper half $C_{n+1} = 1$ and lower half $C_{n+1} = -1$. It can be noticed that the upper half of the truth table of F should be exactly the same as the truth table of f_1 and the lower half should be exactly the same as the truth table of f . Using the values of α_i and β_i we prepare the following model of n -variable PUF

$$\Delta(C)' = \alpha'_0 P_0 + (\alpha'_1 + \beta'_0) P_1 + \dots + (\alpha'_n + \beta'_{n-1}) P_n + \beta'_n, \quad (3.4)$$

with $\alpha'_i = \alpha_i$ for $i = 0, \dots, n$; $\beta'_i = \beta_i$ for $i = 0, \dots, n-1$ and $\beta'_n = (\alpha_{n+1} + \beta_n) + \beta_{n+1}$. The existence of α_i, β_i guarantees that the PUF described in Equation (3.4) will be able to generate the truth table of f_1 . This is a contradiction as $f_1 \notin \mathcal{B}_n^{\text{PUF}}$. Thus $F = f_1 \parallel f \notin \mathcal{B}_{n+1}^{\text{PUF}}$. Similar argument works to prove $F = f \parallel f_1 \notin \mathcal{B}_{n+1}^{\text{PUF}}$. \square

From Lemma 1 and Theorem 2, it is clear that $\mathcal{B}_n^{\text{PUF}} \subset \mathcal{B}_n$ for $n \geq 2$. In fact we can directly say that if $f \in \mathcal{B}_{n+1}^{\text{PUF}}$ then $f = f_1 \parallel f_2$ where $f_1, f_2 \in \mathcal{B}_n^{\text{PUF}}$. With this we would like to investigate $\mathcal{B}_3^{\text{PUF}}$. Proposition 3 claims that $|\mathcal{B}_2^{\text{PUF}}| = 14$. Now if we prepare a 3-variable Boolean function by concatenating these 14 Boolean functions from $\mathcal{B}_2^{\text{PUF}}$ then we can have maximum 196 Boolean functions. The most natural question is that whether all such Boolean functions belong to $\mathcal{B}_3^{\text{PUF}}$ or not. To answer this, we note the following result.

Proposition 4. Consider $f_1 = (1 \ 1 \ 0 \ 1), f_2 = (0 \ 1 \ 0 \ 0) \in \mathcal{B}_2^{\text{PUF}}$ and $f = f_1 \parallel f_2$. The Boolean function $f \notin \mathcal{B}_3^{\text{PUF}}$.

Proof. We construct a truth table of $f = f_1 \parallel f_2$ for a 3-length PUF, where $f_1 = (1 \ 1 \ 0 \ 1), f_2 = (0 \ 1 \ 0 \ 0) \in \mathcal{B}_2^{\text{PUF}}$.

C_3	C_2	C_1	$\Delta(C)$	$f = f_1 \parallel f_2$
1	1	1	$\alpha_1 + (\alpha_2 + \beta_1) + (\alpha_3 + \beta_2) + \beta_3 > 0$	1
1	1	-1	$-\alpha_1 + (\alpha_2 + \beta_1) + (\alpha_3 + \beta_2) + \beta_3 > 0$	1
1	-1	1	$-\alpha_1 - (\alpha_2 + \beta_1) + (\alpha_3 + \beta_2) + \beta_3 < 0$	0
1	-1	-1	$\alpha_1 - (\alpha_2 + \beta_1) + (\alpha_3 + \beta_2) + \beta_3 > 0$	1
-1	1	1	$-\alpha_1 - (\alpha_2 + \beta_1) - (\alpha_3 + \beta_2) + \beta_3 < 0$	0
-1	1	-1	$\alpha_1 - (\alpha_2 + \beta_1) - (\alpha_3 + \beta_2) + \beta_3 > 0$	1
-1	-1	1	$\alpha_1 + (\alpha_2 + \beta_1) - (\alpha_3 + \beta_2) + \beta_3 < 0$	0
-1	-1	-1	$-\alpha_1 + (\alpha_2 + \beta_1) - (\alpha_3 + \beta_2) + \beta_3 < 0$	0

First we consider the following pairs of equations from the above truth table.

$$\begin{cases} -\alpha_1 + (\alpha_2 + \beta_1) + (\alpha_3 + \beta_2) + \beta_3 > 0 \\ \alpha_1 - (\alpha_2 + \beta_1) - (\alpha_3 + \beta_2) + \beta_3 > 0 \end{cases} \quad (3.5)$$

$$\begin{cases} -\alpha_1 - (\alpha_2 + \beta_1) + (\alpha_3 + \beta_2) + \beta_3 < 0 \\ \alpha_1 + (\alpha_2 + \beta_1) - (\alpha_3 + \beta_2) + \beta_3 < 0 \end{cases} \quad (3.6)$$

From Equation (3.5) we get $\beta_3 > 0$ and from Equation (3.6) we get $\beta_3 < 0$. This is a contradiction. Thus $f = f_1 \parallel f_2 \notin \mathcal{B}_3^{\text{PUF}}$. \square

Proposition 4 shows that even if we take any two Boolean functions f_1, f_2 from $\mathcal{B}_2^{\text{PUF}}$ then $f = f_1 \parallel f_2$ may not belong to $\mathcal{B}_3^{\text{PUF}}$. Now given the above results, the first two questions that come to our mind are as follows. Let $f_1, f_2 \in \mathcal{B}_n^{\text{PUF}}$ and $F = f_1 \parallel f_2$.

- Can we have examples of f_1, f_2 , such that $F \in \mathcal{B}_{n+1}^{\text{PUF}}$?
- Can we have examples of f_1, f_2 , such that $F \notin \mathcal{B}_{n+1}^{\text{PUF}}$?

We show that the first case happens when $f_2 = f_1^c$ and the second case is achieved when $f_2 = f_1$, and they are non-constant functions.

Proposition 5. *If $f \in \mathcal{B}_n^{\text{PUF}}$ then $(f \parallel (1 \oplus f)) \in \mathcal{B}_{n+1}^{\text{PUF}}$.*

Proof. It is given that $f \in \mathcal{B}_n^{\text{PUF}}$. As already mentioned, the mathematical model of f is given as $\Delta(C) = \alpha_1 P_0 + (\alpha_2 + \beta_1) P_1 + \dots + (\alpha_n + \beta_{n-1}) P_{n-1} + \beta_n$, with $P_i = \prod_{k=i+1}^n C_k$. Here $C = (C_1, C_2, \dots, C_n)$ is the input of length n and the values of α_i, β_i exist as

$f \in \mathcal{B}_n^{\text{PUF}}$. Now consider the mathematical model of $(n + 1)$ -length PUF and the form the inequalities using the truth values $f \parallel (1 \oplus f)$. Let the mathematical form of that $(n + 1)$ -length PUF be $\alpha'_1 P'_0 + (\alpha'_2 + \beta'_1) P'_1 + \dots + (\alpha'_{n+1} + \beta'_n) P'_n + \beta'_{n+1}$. It can be noticed that here $P'_i = \prod_{k=i+1}^{n+1} C_k$. The form of the system of inequalities will be $M \cdot X$ and the inequality sign corresponding to each row will depend on the corresponding truth value of $f \parallel (1 \oplus f)$. Here $M = [M_1, \dots, M_{2^{n+1}}]^T$, $X = [\alpha'_1, \alpha'_2 + \beta'_1, \dots, \alpha'_{n+1} + \beta'_n, \beta'_{n+1}]$ and $M_j = [P_0, P_1, \dots, P_n, 1]$, the index j corresponds to the j -th element of $\{-1, 1\}^{n+1}$. We will break this system of inequalities into two parts (top and bottom) depending upon the sign of C_{n+1} . In the top part we consider $C_{n+1} = 1$ (as if variable value $x_{n+1} = 0$, and in the bottom part we consider $C_{n+1} = -1$ (variable value $x_{n+1} = 1$, as in the truth table). The form of each row of $M \cdot X$ corresponding to $C_{n+1} = 1$ will be

$$\alpha'_1 P_0 + (\alpha'_2 + \beta'_1) P_1 + \dots + (\alpha'_n + \beta'_{n-1}) P_{n-1} + (\alpha'_{n+1} + \beta'_n) + \beta'_{n+1}. \quad (3.7)$$

Similarly, the mathematical form of the each row of $M \cdot X$ corresponding to $C_{n+1} = -1$ will be

$$-\alpha'_1 P_0 - (\alpha'_2 + \beta'_1) P_1 - \dots - (\alpha'_n + \beta'_{n-1}) P_{n-1} - (\alpha'_{n+1} + \beta'_n) + \beta'_{n+1}. \quad (3.8)$$

Now our aim is to construct the inequalities whose left hand side is $M \cdot X$ and the inequalities sign will be decided from the truth values of $f \parallel (1 \oplus f)$. We also look for a possible solution α'_i, β'_i from these inequalities. It can be noticed that for a fixed value of (C_1, \dots, C_n) , whatever be the inequality sign of the row of $M \cdot X$ corresponding to Equation (3.7), the row of $M \cdot X$ corresponding to Equation (3.8) will have opposite inequality sign as the row of $M \cdot X$ corresponding to Equation (3.7) corresponds to f and the row of $M \cdot X$ corresponding Equation (3.8) corresponds to $(1 \oplus f)$. From the pattern of these inequalities and the known values of α_i, β_i , one can obtain a solution of α'_i, β'_i and the form of that solution is $\alpha'_i = \alpha_i$, for $i = 1, \dots, n$, $\beta'_i = \beta_i$ for $i = 1, \dots, n - 1$, $\beta'_{n+1} = \beta_n$, $(\alpha'_{n+1} + \beta'_n) = 0$. The existence of the solution α'_i, β'_i from values of α_i, β_i proves that if $f \in \mathcal{B}_n^{\text{PUF}}$ then $f \parallel (1 \oplus f)$ will also belong to $\mathcal{B}_{n+1}^{\text{PUF}}$. \square

That is, if $f \in \mathcal{B}_n^{\text{PUF}}$ then $g(x_1, \dots, x_n, x_{n+1}) = (x_{n+1} \oplus f(x_1, \dots, x_n)) \in \mathcal{B}_{n+1}^{\text{PUF}}$ where x_{n+1} is the newly introduced variable. Next let us present a technical result

related to constant functions. Here, by the constant functions we mean that the function output is fixed to either 0 or 1 for all the input values.

Proposition 6. *For every n , the constant functions belong to $\mathcal{B}_n^{\text{PUF}}$.*

Proof. Let the mathematical model of the n -length PUF be $\Delta(C) = \alpha_1 P_0 + (\alpha_2 + \beta_1) P_1 + \cdots + (\alpha_n + \beta_{n-1}) P_{n-1} + (\alpha_n + \beta_{n-1}) P_{n-1} + \beta_n$, where $P_i = \prod_{k=i+1}^n C_k$. From this mathematical model we can construct $M \cdot X$, where $M = [M_1, \dots, M_{2^n}]^T$ and $X = [\alpha_1, \alpha_2 + \beta_1, \dots, \beta_n]$. Here $M_j = [P_0, \dots, P_{n-1}, 1]$ corresponding to j -th element of $\{-1, 1\}^n$. If $C_n = 1$ then we have the following equation in each row of $M \cdot X$ and P'_i will vary with each row.

$$\Delta(C) = \alpha_1 P'_0 + (\alpha_2 + \beta_1) P'_1 + \cdots + (\alpha_{n-1} + \beta_{n-2}) P'_{n-2} + (\alpha_n + \beta_{n-1}) P_{n-1} + \beta_n. \quad (3.9)$$

If $C_n = -1$ then we have the following equation in each row of $M \cdot X$ and P'_i will vary with each row.

$$\Delta(C) = -\alpha_1 P'_0 - (\alpha_2 + \beta_1) P'_1 - \cdots - (\alpha_{n-1} + \beta_{n-2}) P'_{n-2} - (\alpha_n + \beta_{n-1}) P_{n-1} + \beta_n \quad (3.10)$$

Here $P'_i = \prod_{k=i+1}^{n-1} C_k$. To get all 0's in the truth table generated from $\Delta(C)$, we should have $\Delta(C) < 0$ i.e., each row of $M \cdot X$ should be < 0 . Thus from Equations (3.9), (3.10) we obtain,

$$\begin{aligned} \beta_n &< -(\alpha_1 P'_0 + (\alpha_2 + \beta_1) P'_1 + \cdots + (\alpha_{n-1} + \beta_{n-2}) P'_{n-2} + (\alpha_n + \beta_{n-1}) P_{n-1}), \\ \beta_n &< (\alpha_1 P'_0 + (\alpha_2 + \beta_1) P'_1 + \cdots + (\alpha_{n-1} + \beta_{n-2}) P'_{n-2} + (\alpha_n + \beta_{n-1}) P_{n-1}). \end{aligned}$$

Let $K = \max_{\text{rows of } M \cdot X} \{ |(\alpha_1 P'_0 + (\alpha_2 + \beta_1) P'_1 + \cdots + (\alpha_n + \beta_{n-1}) P'_{n-1} + (\alpha_n + \beta_{n-1}))| \}$. If we select $\beta_n < -K$ then each row of $M \cdot X$ will be < 0 , i.e., $\Delta(C)$ will be < 0 for all $C \in \{-1, 1\}^n$. This proves the existence of such α_i, β_i for any n . Hence it provides the existence of the function f in $\mathcal{B}_n^{\text{PUF}}$ for any n , such that $f(x) = 0$ for all $x \in \{0, 1\}^n$. We already know from Lemma 1 that $f \in \mathcal{B}_n^{\text{PUF}}$ if and only if $(1 \oplus f) \in \mathcal{B}_n^{\text{PUF}}$. Thus, both the constant functions belong to $\mathcal{B}_n^{\text{PUF}}$ for every n . \square

Now let us get into the non-existence result.

Proposition 7. *For every non constant function $f \in \mathcal{B}_n$, if $f \in \mathcal{B}_n^{\text{PUF}}$ then $(f \parallel f) \notin \mathcal{B}_{n+1}^{\text{PUF}}$.*

Proof. As in the earlier cases, consider the mathematical model of an $(n + 1)$ -length PUF $\Delta(C) = \alpha'_1 P'_0 + (\alpha'_2 + \beta'_1) P'_1 + \dots + (\alpha'_{n+1} + \beta'_n) P'_{n+1} + \beta'_{n+1}$ where $P_i = \prod_{k=i+1}^{n+1} C_k$ and $C = (C_1, \dots, C_{n+1})$. We first prepare the matrix $M = [M_1, \dots, M_{2^{n+1}}]^T$ where $M_j = [P_0, \dots, P_n, 1]$ and it corresponds to j -th element of $\{-1, 1\}^{n+1}$. After that construct $M \cdot X$ where $X = [\alpha'_1, \alpha'_2 + \beta'_1, \dots, \beta'_{n+1}]^T$. Using $M \cdot X$ we construct a system of inequalities where inequality sign corresponding to each row of $M \cdot X$ will be decided from the truth values of $f \parallel f$ ($f \in \mathcal{B}_n^{\text{PUF}}$). For $C_{n+1} = 1$ and -1 the mathematical forms of the respective rows of $M \cdot X$ are given in Equation (3.11), (3.12) respectively.

$$\alpha'_1 P_0 + (\alpha'_2 + \beta'_1) P_1 + \dots + (\alpha'_n + \beta'_{n-1}) P_n + (\alpha'_{n+1} + \beta'_n) + \beta'_{n+1} \quad (3.11)$$

$$-\alpha'_1 P_0 - (\alpha'_2 + \beta'_1) P_1 - \dots - (\alpha'_n + \beta'_{n-1}) P_n - (\alpha'_{n+1} + \beta'_n) + \beta'_{n+1} \quad (3.12)$$

Assume in the truth values of f , there is 1 in i -th position and 0 in j -th position, i.e., f is not constant. Corresponding to 1 in i -th position we will have following two inequalities from the i -th row of the matrix $M \cdot X$.

$$\begin{aligned} \alpha'_1 P_0 + (\alpha'_2 + \beta'_1) P_1 + \dots + (\alpha'_n + \beta'_{n-1}) P_n + (\alpha'_{n+1} + \beta'_n) + \beta'_{n+1} &> 0, \\ -\alpha'_1 P_0 - (\alpha'_2 + \beta'_1) P_1 - \dots - (\alpha'_n + \beta'_{n-1}) P_n - (\alpha'_{n+1} + \beta'_n) + \beta'_{n+1} &> 0. \end{aligned}$$

This implies $\beta'_{n+1} > 0$. Now corresponding to 0 in j -th position we will have two more inequalities from the j -th row of the matrix $M \cdot X$.

$$\begin{aligned} \alpha'_1 P_0 + (\alpha'_2 + \beta'_1) P_1 + \dots + (\alpha'_n + \beta'_{n-1}) P_n + (\alpha'_{n+1} + \beta'_n) + \beta'_{n+1} &< 0, \\ -\alpha'_1 P_0 - (\alpha'_2 + \beta'_1) P_1 - \dots - (\alpha'_n + \beta'_{n-1}) P_n - (\alpha'_{n+1} + \beta'_n) + \beta'_{n+1} &< 0. \end{aligned}$$

This implies $\beta'_{n+1} < 0$. Here one pair of inequalities are providing $\beta'_{n+1} > 0$ and other pair of inequalities give $\beta'_{n+1} < 0$, which is a contradiction. Thus system of inequalities formed from $M \cdot X$ and $f \parallel f$ is not solvable. Thus if $f \in \mathcal{B}_n^{\text{PUF}}$ and f is non-constant, then $(f \parallel f) \notin \mathcal{B}_{n+1}^{\text{PUF}}$. \square

We summarize the values of $|\mathcal{B}_n^{\text{PUF}}|$ for different values of n in Table 3.2.

For higher values of n , we have considered the mathematical model of PUF described in Equation 2.1 for different values of n and exhaustively searched the

n	$ \mathcal{B}_n^{\text{PUF}} $
1	4
2	14
3	104
4	1882

Table 3.2: $|\mathcal{B}_n^{\text{PUF}}|$ for different n

Boolean functions which belong to $\mathcal{B}_n^{\text{PUF}}$. For $n = 3, 4$ we have observed that $|\mathcal{B}_3^{\text{PUF}}| = 104 < |\mathcal{B}_2^{\text{PUF}}|^2 - |\mathcal{B}_2^{\text{PUF}}| + 2$ and $|\mathcal{B}_4^{\text{PUF}}| = 1882 < |\mathcal{B}_3^{\text{PUF}}|^2 - |\mathcal{B}_3^{\text{PUF}}| + 2$. From this, the following result follows.

Theorem 3. *For any value of n , $|\mathcal{B}_{n+1}^{\text{PUF}}| \leq |\mathcal{B}_n^{\text{PUF}}|^2 - |\mathcal{B}_n^{\text{PUF}}| + 2$. Further, for $n \geq 4$, $\frac{|\mathcal{B}_n^{\text{PUF}}|}{|\mathcal{B}_n|} < \frac{1}{2^{5 \cdot 2^{n-4}}}$.*

Proof. The first result follows from Theorem 2. The next result is initiated from exhaustive experiments, where for different values of delay parameters we have observed that $|\mathcal{B}_4^{\text{PUF}}| = 1882$. Regarding the exhaustive experiment supporting the proof we refer to Algorithm 2 below. If we compute $\frac{|\mathcal{B}_4^{\text{PUF}}|}{|\mathcal{B}_4|} = \frac{1882}{2^{24}} < \frac{1}{2^5} = \frac{1}{2^{5 \cdot 2^{4-4}}}$. Assume that the relation holds for $n = k$, for some $k > 4$, i.e., $\frac{|\mathcal{B}_k^{\text{PUF}}|}{|\mathcal{B}_k|} < \frac{1}{2^{5 \cdot 2^{k-4}}}$. For $n = k + 1$, following Theorem 2 we have,

$$\frac{|\mathcal{B}_{k+1}^{\text{PUF}}|}{|\mathcal{B}_{k+1}|} \leq \frac{|\mathcal{B}_k^{\text{PUF}}|^2}{|\mathcal{B}_k|^2} < \left(\frac{1}{2^{5 \cdot 2^{k-4}}}\right)^2 = \left(\frac{1}{2^{5 \cdot 2^{(k+1)-4}}}\right) \quad (3.13)$$

Hence for $n \geq 4$, $\frac{|\mathcal{B}_n^{\text{PUF}}|}{|\mathcal{B}_n|} < \frac{1}{2^{5 \cdot 2^{n-4}}}$. □

Although the bound derived in Theorem 3 is not tight, it provides a significant estimation about $\mathcal{B}_n^{\text{PUF}}$. Now the question is how one can obtain $\mathcal{B}_{n+1}^{\text{PUF}}$ exhaustively. One informal way is, consider large number of values varying the delay parameters to construct $(n+1)$ variable PUFs and enumerate the number of distinct ones. However, this cannot be used as a proof but it can be used to find the set $\mathcal{B}_n^{\text{PUF}}$ for small values

of n .

Algorithm 2: Construction of $\mathcal{B}_{n+1}^{\text{PUF}}$ from $\mathcal{B}_n^{\text{PUF}}$

Input : $\mathcal{B}_n^{\text{PUF}}$
Output: $\mathcal{B}_{n+1}^{\text{PUF}}$

- 1 Assign $\Delta(C) = \alpha_1 P_0 + (\alpha_2 + \beta_1) P_1 + \dots + (\alpha_{n+1} + \beta_n) P_n + \beta_{n+1}$, $P_i = \prod_{k=i+1}^n C_k$;
- 2 **for** each $f_i \in \mathcal{B}_n^{\text{PUF}}$ **do**
- 3 $F_1 = \{\}$;
- 4 **if** $C_{n+1} = 1$ **then**
- 5 **if** $f_i(C_1, \dots, C_n) = 1$ **then**
- 6 Construct equation $\Delta(C) > 0$ and include $\Delta(C) > 0$ in F_1 ;
- 7 **end**
- 8 **else**
- 9 Construct equation $\Delta(C) < 0$ and include $\Delta(C) < 0$ in F_1 ;
- 10 **end**
- 11 **end**
- 12 **for** each $f_j \in \mathcal{B}_n^{\text{PUF}}$ **do**
- 13 $F_2 = \{\}$;
- 14 **if** $C_{n+1} = -1$ **then**
- 15 **if** $f_j(C_1, \dots, C_n) = 1$ **then**
- 16 Construct equation $\Delta(C) > 0$ and include $\Delta(C) > 0$ in F_2 ;
- 17 **end**
- 18 **else**
- 19 Construct equation $\Delta(C) < 0$ and include $\Delta(C) < 0$ in F_2 ;
- 20 **end**
- 21 **end**
- 22 **if** $F = F_1 \cup F_2$ is solvable **then**
- 23 Construct $f = f_1 \parallel f_2$ and include f in $\mathcal{B}_{n+1}^{\text{PUF}}$;
- 24 **end**
- 25 **end**
- 26 **end**
- 27 **return** $\mathcal{B}_{n+1}^{\text{PUF}}$;

Here we provide an iterative way of completely enumerating $\mathcal{B}_{n+1}^{\text{PUF}}$ from $\mathcal{B}_n^{\text{PUF}}$.

In Algorithm 2 we consider the mathematical model of $(n + 1)$ -variable PUF, i.e., $\Delta(C) = \alpha_1 P_0 + (\alpha_2 + \beta_1) P_1 + \dots + (\alpha_{n+1} + \beta_n) P_n + \beta_{n+1}$, $P_i = \prod_{k=i+1}^n C_k$. That is $\Delta(C)$ can be considered as a Boolean function on $C = (C_1, C_2, \dots, C_{n+1})$, the challenge inputs corresponding to $(n + 1)$ -length PUF. Consider any two $f_1, f_2 \in \mathcal{B}_n^{\text{PUF}}$. Let $f = f_1 \parallel f_2$. For $C_{n+1} = 1$ we prepare the system of inequalities involving α_i, β_i , based on the truth table of f_1 . Similarly, for $C_{n+1} = -1$ we construct the system of inequalities involving α_i, β_i based on the truth table of f_2 . If this system of equations is solvable then we include the Boolean function f in $\mathcal{B}_{n+1}^{\text{PUF}}$ which corresponds to the $(n + 1)$ -length PUF $\Delta(C)$. If we continue this process for all $f_1, f_2 \in \mathcal{B}_n^{\text{PUF}}$ then we will have $\mathcal{B}_{n+1}^{\text{PUF}}$.

We have implemented Algorithm 2 in SageMath 9.2 [sag] and enumerated $\mathcal{B}_{n+1}^{\text{PUF}}$ for $n = 1, 2, 3$. Algorithm 2 outputs the correct set $\mathcal{B}_{n+1}^{\text{PUF}}$ in 1.891 sec, 72.320 sec and 2553.546 sec for $n = 1, 2, 3$ respectively. For $n = 1, 2$ we have run the experiment in a laptop with processor of 2.80 GHz clock, 16 GB RAM and Linux (Ubuntu 20.04.03) environment. For $n = 3$ we have used multiprocessing in our implementation and the program was executed in a high performance computing machine with processor of 2.30 GHz clock, 72 CPUs, 96 GB RAM and Linux (CentOS 7) environment.

3.4 Determining whether $f \in \mathcal{B}_n^{\text{PUF}}$

In Section 3.3, it has been observed that truth table of certain class of n -variable ($n > 1$) Boolean function can not be generated from an n -length PUF. While enumerating the functions in $\mathcal{B}_n^{\text{PUF}}$, a pair (we show in Proposition 7 that they must be distinct) has been chosen from $\mathcal{B}_{n-1}^{\text{PUF}}$ and the concatenated function has been studied above. We now want to check whether the truth table of the new function belongs to $\mathcal{B}_n^{\text{PUF}}$ by solving inequalities formed using the compact form $M \cdot X$ and the truth values. If there is a solution, then it belongs to $\mathcal{B}_n^{\text{PUF}}$, else not. Thus, this strategy can be used as a distinguisher to identify whether a Boolean function belongs to $\mathcal{B}_n^{\text{PUF}}$ or not. Let us first formalize this.

Suppose a truth table of an n -variable Boolean function f is provided. With these truth values we will determine whether this truth table is generated using an n -length PUF or not, i.e., whether $f \in \mathcal{B}_n^{\text{PUF}}$ or $f \notin \mathcal{B}_n^{\text{PUF}}$. To check this we first consider the mathematical model of an n -length PUF, which is $\Delta(C) = \alpha_1 P_0 + (\alpha_2 + \beta_1) P_1 + \dots +$

$(\alpha_n + \beta_{n-1})P_n + \beta_n$, with $P_i = \prod_{k=i+1}^n C_k$ and $C = (C_1, C_2, \dots, C_n)$ is n -length input. We first prepare the equations of the form $M \cdot X$ by considering all $C \in \{-1, 1\}^n$, where $M = [M_1, \dots, M_{2^n}]^T$, $X = [\alpha_1, \alpha_2 + \beta_1, \dots, \beta_n]^T$ and the j -th row of M is $M_j = [P_0, \dots, P_{n-1}, 1]$. Here j -th row corresponds to the j -th element of $\{-1, 1\}^n$. From the truth values of f and $M \cdot X$, we can prepare a system of inequalities. For input C (say j -th element of $\{-1, 1\}^n$) if the truth value is 1 then we assign > 0 inequality to the j -th row of $M \cdot X$ and if the truth value is 0 then we assign < 0 to the j -th row of $M \cdot X$. In this way we can prepare a system of 2^n inequalities involving α_i, β_i , for $i = 1, \dots, n$. If this system of equations has non trivial solution then $f \in \mathcal{B}_n^{\text{PUF}}$ else $f \notin \mathcal{B}_n^{\text{PUF}}$. We present this deterministic technique in Algorithm 3.

Algorithm 3: Deterministic Distinguisher on PUF

Input : Truth table T of $f \in \mathcal{B}_n$
Output: $f \in \mathcal{B}_n^{\text{PUF}}$ or $f \notin \mathcal{B}_n^{\text{PUF}}$

- 1 Consider $\Delta(C) = \alpha_1 P_0 + (\alpha_2 + \beta_1)P_1 + \dots + (\alpha_n + \beta_{n-1})P_n + \beta_n$, with

$$P_i = \prod_{k=i+1}^n C_k \text{ and } C = (C_1, C_2, \dots, C_n);$$
- 2 Construct $M \cdot X$ by considering all $C \in \{-1, 1\}^n$, where

$$M = [M_1, \dots, M_{2^n}]^T, X = [\alpha_1, \alpha_2 + \beta_1, \dots, \beta_n]^T$$
 and the j -th row of M , $M_j = [P_0, \dots, P_{n-1}, 1]$ corresponds to the j -th element of $\{-1, 1\}^n$;
- 3 **for** Each $C \in \{-1, 1\}^n$ **do**
 - 4 | **if** $f(C) = 1$ **then**
 - 5 | | Assign > 0 to the respective row of $M \cdot X$;
 - 6 | **end**
 - 7 | **else**
 - 8 | | Assign < 0 to the respective row of $M \cdot X$;
 - 9 | **end**
- 10 **end**
- 11 $S =$ non trivial solution of α_i, β_i from the above inequalities;
- 12 **if** $S \neq \{\}$ **then**
- 13 | $R \leftarrow f \in \mathcal{B}_n^{\text{PUF}}$;
- 14 **end**
- 15 **else**
- 16 | $R \leftarrow f \notin \mathcal{B}_n^{\text{PUF}}$;
- 17 **end**
- 18 **return** R ;

We have run a few experiments in SageMath9.2 [sag] in a laptop with processor of 2.80 GHz clock, 16 GB RAM and Linux (Ubuntu 20.04.3) environment to check the correctness of the Algorithm 3 for different values of n and randomly chosen $f \in \mathcal{B}_n$.

Experimental observations are presented in Table 3.3.

n	f	Decision	Time Required (in sec)
2	$f^{(2)} = 5$	$f^{(2)} \notin \mathcal{B}_2^{\text{PUF}}$	0.767
2	$f^{(2)} = 4$	$f^{(2)} \in \mathcal{B}_2^{\text{PUF}}$	0.791
3	$f^{(3)} = 4A$	$f^{(3)} \notin \mathcal{B}_3^{\text{PUF}}$	1.014
3	$f^{(3)} = 4B$	$f^{(3)} \in \mathcal{B}_3^{\text{PUF}}$	1.280
4	$f^{(4)} = BECF$	$f^{(4)} \notin \mathcal{B}_4^{\text{PUF}}$	2.789
4	$f^{(4)} = BEC3$	$f^{(4)} \in \mathcal{B}_4^{\text{PUF}}$	3.535
5	$f^{(5)} = 00047454$	$f^{(5)} \notin \mathcal{B}_5^{\text{PUF}}$	47.495
6	$f^{(6)} = 00000000056A739$	$f^{(6)} \notin \mathcal{B}_6^{\text{PUF}}$	448.251

Table 3.3: Experimental validation of Algorithm 3. The functions are written in hexadecimal format.

The system of inequalities generated in Algorithm 3 contains 2^n many inequalities involving $2n$ number of variables. This is an over-defined system of inequalities. We can introduce a new variable corresponding to each inequality and select the sign of the new variable in such a way that the corresponding inequality becomes an equality. This process will require 2^n number of new variables. This final system now becomes a system of 2^n linear equations involving $(2n + 2^n)$ variables. Solving such system will trivially require $\Omega(2n + 2^n)$ complexity and if the solution exists then it will have infinitely many solutions.

From the structure of the inequalities one can observe that only the sign of the inequalities changes with the truth values of any function. The structure of left hand side of each inequality remains the same. To understand this in a better way we present the form of the inequalities for one particular truth table each for $n = 3$ and $n = 4$.

$$\text{For } n = 3, M = \begin{bmatrix} 1 & 1 & 1 & 1 \\ -1 & 1 & 1 & 1 \\ -1 & -1 & 1 & 1 \\ 1 & -1 & 1 & 1 \\ -1 & -1 & -1 & 1 \\ 1 & -1 & -1 & 1 \\ 1 & 1 & -1 & 1 \\ -1 & 1 & -1 & 1 \end{bmatrix}, X = \begin{bmatrix} \alpha_1 \\ \alpha_2 + \beta_1 \\ \alpha_3 + \beta_2 \\ \beta_3 \end{bmatrix} \text{ and } M \cdot X \begin{bmatrix} > 0 \\ > 0 \\ > 0 \\ > 0 \\ > 0 \\ < 0 \\ > 0 \\ < 0 \end{bmatrix}$$

$$\text{For } n = 4, M = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ -1 & 1 & 1 & 1 & 1 \\ -1 & -1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 & 1 \\ -1 & -1 & -1 & 1 & 1 \\ 1 & 1 & -1 & 1 & 1 \\ -1 & 1 & -1 & 1 & 1 \\ -1 & -1 & -1 & -1 & 1 \\ 1 & -1 & -1 & -1 & 1 \\ 1 & 1 & -1 & -1 & 1 \\ -1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & -1 & 1 \\ -1 & 1 & 1 & -1 & 1 \\ -1 & -1 & 1 & -1 & 1 \\ 1 & -1 & 1 & -1 & 1 \end{bmatrix}, X = \begin{bmatrix} \alpha_1 \\ \alpha_2 + \beta_1 \\ \alpha_3 + \beta_2 \\ \alpha_3 + \beta_3 \\ \beta_4 \end{bmatrix} \text{ and } M \cdot X \begin{bmatrix} > 0 \\ > 0 \\ > 0 \\ > 0 \\ > 0 \\ < 0 \\ > 0 \\ < 0 \\ < 0 \\ < 0 \\ < 0 \\ < 0 \\ < 0 \\ < 0 \\ < 0 \\ < 0 \\ < 0 \\ < 0 \\ < 0 \\ < 0 \end{bmatrix}$$

From these equations one can observe that the structure of the above system of equations do not change with the increasing values of n . Only the number of variables are added with the increment of n and the inequality sign changes with the truth table of the function. Given the structure of the inequalities, it is worth exploring whether such systems can be analysed in an efficient manner. However, it is clear that studying the truth table will require $\Omega(2^n)$ operations.

3.4.1 Improved but randomized technique

It is already known to us that for a given truth table of $f \in \mathcal{B}_n$ if the system of inequalities do not have solution for α_i, β_i then $f \notin \mathcal{B}_n^{\text{PUF}}$. However, we already have the result from Theorem 2 that if $f \in \mathcal{B}_{n+1}^{\text{PUF}}$ with $f = f_1 \parallel f_2$ then $f_1, f_2 \in \mathcal{B}_n^{\text{PUF}}$. Thus, we can divide the problem in smaller parts given $f \in \mathcal{B}_{n+1}$. If at least one of f_1 or f_2 does not belong to $\mathcal{B}_n^{\text{PUF}}$, we can certainly deduce that $f \notin \mathcal{B}_{n+1}^{\text{PUF}}$. However, if both belong to $\mathcal{B}_n^{\text{PUF}}$, then we cannot answer with certainty. However, as most of the Boolean functions do not belong to $\mathcal{B}_n^{\text{PUF}}$, this provides a very fast practical method to start with. We will thus outline such a method here.

Note that all the functions of $\mathcal{B}_4^{\text{PUF}}$ are enumerated in Section 3.3. Let us consider that all the functions of $\mathcal{B}_u^{\text{PUF}}$ are enumerated for $u \geq 4$ and stored in an indexed database \mathcal{D} . Now consider a function $f(x_1, \dots, x_u, x_{u+1}, \dots, x_n) \in \mathcal{B}_n$. One can fix the variables x_{u+1}, \dots, x_n to a fixed value and then obtain the truth table of an u -variable function g . Now one can directly test using the solutions of inequalities whether $g \in \mathcal{B}_u^{\text{PUF}}$. The same testing can be done by efficiently by searching through \mathcal{D} .

Let us now present an example. Let $f \in \mathcal{B}_4$, and $f = f_1 \parallel f_2$, where $f_1, f_2 \in \mathcal{B}_3$. We know that $|\mathcal{B}_3^{\text{PUF}}| = 104$ and $|\mathcal{B}_4^{\text{PUF}}| = 1882$. Now if we select f from \mathcal{B}_4 uniformly at random, then $Pr[f \in \mathcal{B}_4^{\text{PUF}}] = \frac{1882}{2^{24}}$. Now consider that we check with certain algorithm that $f_1, f_2 \in \mathcal{B}_3^{\text{PUF}}$. Given our constraint in Theorem 3, the number of such functions are $104 \times 104 - 104 + 2 = 10714$. Among these options, only 1882 many functions actually belong to $\mathcal{B}_4^{\text{PUF}}$. Thus we obtain the following.

$$Pr[f \in \mathcal{B}_4^{\text{PUF}} | f_1, f_2 \in \mathcal{B}_3^{\text{PUF}}] = \frac{1882}{10714} > \frac{1882}{2^{24}} = Pr[f \in \mathcal{B}_4^{\text{PUF}}]$$

One may note that $\frac{1882}{10714} > 0.175$, whereas $\frac{1882}{2^{24}} < 0.029$, an increase of more than six times. Hence for any $f \in \mathcal{B}_4$, if we note that both $f_1, f_2 \in \mathcal{B}_3^{\text{PUF}}$, then the probability that $f \in \mathcal{B}_4^{\text{PUF}}$ increases significantly. We generalize this below.

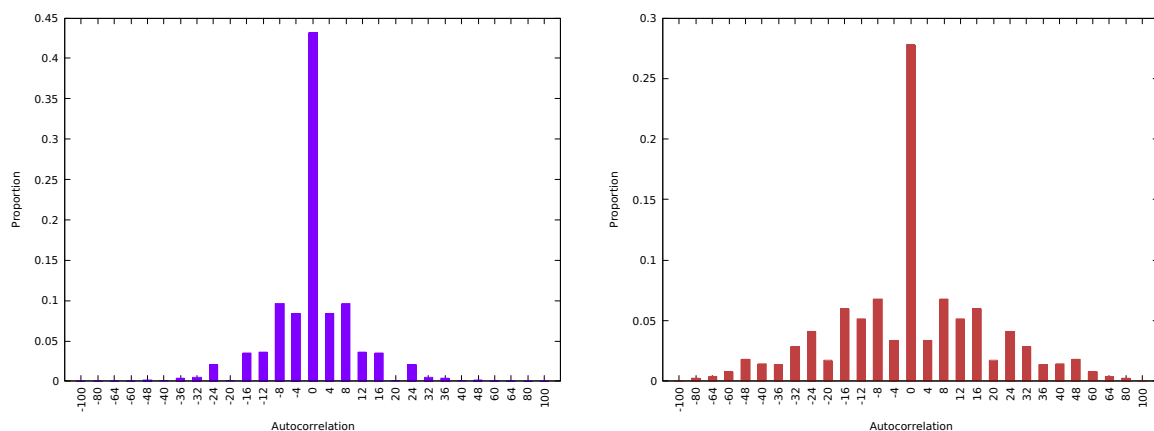
Let $f \in \mathcal{B}_n$ and $Pr[f \in \mathcal{B}_n^{\text{PUF}}] = \frac{|\mathcal{B}_n^{\text{PUF}}|}{|\mathcal{B}_n|}$. We have the set $\mathcal{B}_u^{\text{PUF}}$ for $u < n$. Using this and our search algorithm we first take random 2^u portion of the truth table of f by fixing x_{u+1}, \dots, x_n . Say, we observe that for all possible options of x_{u+1}, \dots, x_n , the generated u -variable functions are present in $\mathcal{B}_u^{\text{PUF}}$. Then,

$$Pr[f \in \mathcal{B}_n^{\text{PUF}} | f_i \in \mathcal{B}_u^{\text{PUF}}, i = 1, \dots, 2^{n-u}] = \frac{|\mathcal{B}_n^{\text{PUF}}|}{|\mathcal{B}_u^{\text{PUF}}|^{2^{n-u}}} > \frac{|\mathcal{B}_n^{\text{PUF}}|}{|\mathcal{B}_n|} = Pr[f \in \mathcal{B}_n^{\text{PUF}}]. \quad (3.14)$$

In the other direction, when any such u -variable section of the n -variable function does not belong to $\mathcal{B}_u^{\text{PUF}}$, then we can immediately deduce that the n -variable function in hand does not belong to $\mathcal{B}_n^{\text{PUF}}$. One may have a look at Table 3.3, and note that solving the inequalities take much higher time as the number of variables increase. Thus, it is much logical to go for the tests in lower number of variables for all practical purposes.

3.5 On Restricted Autocorrelation of Arbiter PUF

In Section 3.2 we have seen that the distribution of Boolean functions and PUFs differs significantly in terms of autocorrelation spectrum. This happens due to the fact that the PUFs depend on multiple device specific parameters and $\mathcal{B}_n^{\text{PUF}} \subset \mathcal{B}_n$ for $n > 2$. Interestingly, if we consider the challenge inputs from $E_{n,k}$ with certain restrictions, then the autocorrelation distributions of random Boolean functions and PUFs become quite close. For measuring this we need to revisit the definition of restricted autocorrelation [see Definition 2.1.1].



(a) Distribution of the Boolean functions in the restricted domain $E_{6,3}$ (b) Distribution of the randomly chosen PUFs in the restricted domain $E_{6,3}$

Figure 3-3: Comparison of the restricted autocorrelation.

As we have discussed, f is an n -variable Boolean function. S_1 and S_2 are two sets defined as $S_1 = \{\mathbf{x} \in E_{n,k} \mid u\text{-th bit of } \mathbf{x} \text{ is } -1\}$, $S_2 = \{\mathbf{x} \in E_{n,k} \mid u\text{-th bit of } \mathbf{x} \text{ is } 1\}$. Note that $E_{n,k} = S_1 \cup S_2$ and $S_1 \cap S_2 = \emptyset$. The restricted autocorrelation of f over $E_{n,k}$ is defined as

$$\mathcal{A}_f^{E_{n,k}} = \sum_{\mathbf{x}_1 \in S_1, \mathbf{x}_2 \in S_2} (-1)^{f(\mathbf{x}_1) \oplus f(\mathbf{x}_2)}.$$

Let us explain the scenario for restricted autocorrelation over the domain $E_{6,3}$. We have classified all the $2^{\binom{6}{3}}$ patterns and computed the distribution of Boolean function corresponding to different restricted autocorrelation values in Figure 3-3. Such autocorrelation values are $\{-100, -80, -64, -60, -48, -40, -36, -32, -24, -20, -16, -12, -8, -4, 0, 4, 8, 12, 16, 20, 24, 32, 36, 40, 48, 60, 64, 80, 100\}$. The

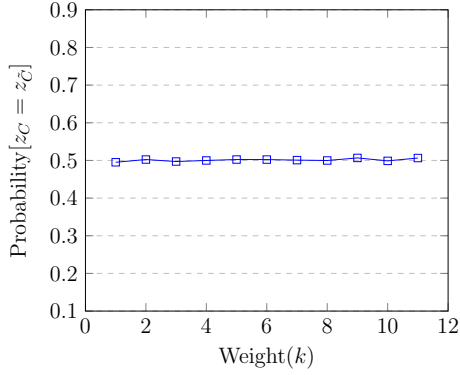


Figure 3-4: Distribution of $Pr[z_C = z_{\tilde{C}}]$ in $E_{12,k}$

Weight(k)	$Pr[z_C = z_{\tilde{C}}]$
1	0.495117
2	0.502397
3	0.497111
4	0.499929
5	0.502405
6	0.502307
7	0.500808
8	0.499840
9	0.506651
10	0.498867
11	0.506392

Table 3.4: $Pr[z_C = z_{\tilde{C}}]$ in $E_{12,k}$

frequency of all such functions are normalized by dividing with $2^{\binom{6}{3}}$. For 6-length PUFs we have randomly searched with 2^{20} different sets of delay parameters (α_i, β_i) and obtained 14100 such distinct functions. For them we also obtained the same set of distinct autocorrelation values. The normalized frequency distribution is drawn in Figure 3-3. A few blocks corresponding to certain autocorrelation values (such as $-100, -80, 80, 100$) in Figure 3-3 are not visible due to very small proportion.

From Figures 3-2 and 3-3, it can be observed that the restricted autocorrelation distribution of the PUFs demonstrates same behavior as the set of Boolean functions. That is the differential characteristics related to the bias is not observed for this restricted domain. That is, if the choice of two distinct challenge pairs can be restricted over certain domains (here one from S_1 and another from S_2 given a specific input bit location u), then the cryptographic weakness related to the bias might be avoided. Other than this different larger classes should be explored where such improved properties can be observed. Very simple model of Arbiter PUFs can be used there as cryptographic components with better confidence.

3.5.1 Theoretical Analysis

We now consider an Arbiter PUF whose inputs are from $E_{n,k}$. Let us divide the complete set $E_{n,k}$ into two subsets S_1 and S_2 , where $S_1 = \{\mathbf{x} : \text{MSB of } \mathbf{x} \text{ is } -1\}$ and $S_2 = \{\mathbf{x} : \text{MSB of } \mathbf{x} \text{ is } 1\}$, i.e., the selected input bit is $u = n$.

Consider challenge input C from S_1 and \tilde{C} from S_2 . For a randomly chosen PUF,

let us denote z_C as the output corresponding to C and $z_{\tilde{C}}$ as the output corresponding to \tilde{C} . Compute the difference $z_C \oplus z_{\tilde{C}}$ for $C \in S_1$ and $\tilde{C} \in S_2$. If we calculate the average for all the points $C \in S_1$ and $\tilde{C} \in S_2$, then we can estimate the quantity $p_i = Pr[z_C = z_{\tilde{C}}]$. Here p_i denotes the probability corresponding to i -th PUF say. We compute the average of all these probabilities (p_i 's) for of all the different cases $E_{12,1}, E_{12,2}, \dots, E_{12,11}$. The obtained experimental data is presented in Table 3.4 and the distribution is plotted in Figure 3-4. Note that $E_{12,0}$ and $E_{12,12}$ are not considered here as $|E_{12,0}| = |E_{12,12}| = 1$. From this experiment, we observe that the average probability is close to 0.5 for all weights $k = 1, \dots, 11$. During the experiments, we have also observed that these probabilities do not depend on the choice of input bit t .

We note that this average probability is very close to 0.5 and that motivates us to explore the following theoretical result.

Theorem 4. *Expectation of $\mathcal{A}_f^{E_{n,k}}$ is equal to $\frac{1}{2}$ for $f \in \mathcal{B}_n^{\text{PUF}}$.*

Proof. Consider two distinct challenge inputs $C, \tilde{C} \in E_{n,k}$ such that they must differ at location t_1 . Here C and \tilde{C} are of the same weight k , hence they will definitely differ at more than one location. Let the m locations where C and \tilde{C} differ be t_1, t_2, \dots, t_m .

Let $\alpha = (\alpha_{t_1+1} + \beta_{t_1})P_{t_1} + (\alpha_{t_1+2} + \beta_{t_1+1})P_{t_1+1} + \dots + (\alpha_{t_2} + \beta_{t_2-1})P_{t_2-1} + (\alpha_{t_3+1} + \beta_{t_3})P_{t_3} + \dots + (\alpha_{t_4} + \beta_{t_4-1})P_{t_4-1} + \dots + (\alpha_{t_m} + \beta_{t_m-1})P_{t_m-1}$ and $X = \Delta(C) - \alpha$. Thus the sign of $\Delta(C)$ corresponding to two challenge inputs C, \tilde{C} will be same if and only if $|\frac{\alpha}{X}| < 1$. Hence the output bits corresponding two inputs C and \tilde{C} will be same if and only if $|\frac{\alpha}{X}| < 1$.

As $\alpha_i, \beta_i \sim \mathcal{N}(0, \sigma)$, the quantity α will follow $\mathcal{N}(0, \sigma_\alpha)$ and X will follow $\mathcal{N}(0, \sigma_X)$, where

$$\sigma_\alpha = \sigma \sqrt{2[(t_2 - t_1) + (t_4 - t_3) + \dots + (t_m - t_{m-1})]}$$

and

$$\sigma_X = \sigma \sqrt{2n - 2[(t_2 - t_1) + (t_4 - t_3) + \dots + (t_m - t_{m-1})]}$$

The probability density functions of α and X will be $f_\alpha(y) = \frac{1}{\sqrt{2\pi\sigma_\alpha}} e^{-\frac{y^2}{2\sigma_\alpha^2}}$, $-\infty < y < \infty$ and $f_X(y) = \frac{1}{\sqrt{2\pi\sigma_X}} e^{-\frac{y^2}{2\sigma_X^2}}$, $-\infty < y < \infty$ respectively. Now we consider

$Y_1 = \frac{\alpha}{X}$ and $Y_2 = X$. So $\alpha = Y_1 Y_2$. The joint distribution of α, X will be $f_{\alpha, X}(\alpha, x) = \frac{1}{2\pi\sigma_\alpha\sigma_X} e^{-\left(\frac{\alpha^2}{2\sigma_\alpha^2} + \frac{x^2}{2\sigma_X^2}\right)}$. Similarly, the joint distribution of Y_1, Y_2 will be $f_{Y_1, Y_2}(y_1, y_2) = \frac{1}{2\pi\sigma_\alpha\sigma_X} e^{-\left(\frac{y_1^2 y_2^2}{2\sigma_\alpha^2} + \frac{y_2^2}{2\sigma_X^2}\right)}$, where $-\infty < y_1, y_2 < \infty$. The distribution of Y_1 will be $f_{Y_1}(y_1) = \int_{-\infty}^{\infty} f_{Y_1, Y_2}(y_1, y_2) dy_2 = \int_{-\infty}^{\infty} \frac{1}{2\pi\sigma_\alpha\sigma_X} e^{-\left(\frac{y_1^2 y_2^2}{2\sigma_\alpha^2} + \frac{y_2^2}{2\sigma_X^2}\right)} y_2 dy_2 = \frac{1}{\pi} \frac{\frac{\sigma_\alpha}{\sigma_X}}{y_1^2 + \left(\frac{\sigma_\alpha}{\sigma_X}\right)^2}$, where $-\infty < y_1 < \infty$.

We already know that the output bits corresponding to the two inputs C and \tilde{C} will be the same if and only if $|\frac{\alpha}{X}| < 1$. To calculate $Pr[|\frac{\alpha}{X}| < 1]$ we need to calculate $Pr[|Y_1| < 1]$.

$$\begin{aligned}
Pr[|Y_1| < 1] &= \left| \int_{-1}^1 \frac{1}{\pi} \frac{\frac{\sigma_\alpha}{\sigma_X}}{y_1^2 + \left(\frac{\sigma_\alpha}{\sigma_X}\right)^2} dy_1 \right| \\
&= \frac{1}{\pi} \left| \left\{ \tan^{-1} \left(\frac{1}{\frac{\sigma_\alpha}{\sigma_X}} \right) - \tan^{-1} \left(\frac{-1}{\frac{\sigma_\alpha}{\sigma_X}} \right) \right\} \right| \\
&= 1 - \frac{2}{\pi} \tan^{-1} \left(\frac{\sigma_\alpha}{\sigma_X} \right) \\
&= 1 - \frac{2}{\pi} \tan^{-1} \left(\sqrt{\frac{(t_2 - t_1) + (t_4 - t_3) + \dots + (t_m - t_{m-1})}{n - [(t_2 - t_1) + (t_4 - t_3) + \dots + (t_m - t_{m-1})]}} \right) \\
&= 1 - \frac{2}{\pi} \tan^{-1} \sqrt{\frac{t}{n - t}}.
\end{aligned}$$

Here $t = (t_2 - t_1) + (t_4 - t_3) + \dots + (t_m - t_{m-1})$. Note that we have selected two distinct challenge inputs C, \tilde{C} from $E_{n,k}$ with the condition that C and \tilde{C} must differ at location t_1 . Without loss of generality, we can assume that t_1 -th location of C has -1 and t_1 -th location of \tilde{C} has 1 . Let $S_1 = \{\mathbf{x} \mid \mathbf{x} \in E_{n,k} \text{ and } t_1\text{-th location of } \mathbf{x} \text{ has } -1\}$, $S_2 = \{\mathbf{x} \mid \mathbf{x} \in E_{n,k} \text{ and } t_1\text{-th location of } \mathbf{x} \text{ has } 1\}$. That is $C \in S_1, \tilde{C} \in S_2$ and we have already noted $|S_1| = \binom{n-1}{k-1}, |S_2| = \binom{n-1}{k}$. If we consider the average probability for all choices of $C \in S_1$ and $\tilde{C} \in S_2$ then we will get the expectation of $\mathcal{A}_f^{E_{n,k}}$, where f is an n -length Arbiter PUF chosen uniformly at random. Hence,

$$\begin{aligned}
\text{Expectation of } \mathcal{A}_f^{E_{n,k}} &= \frac{1}{\binom{n-1}{k} \times \binom{n-1}{k-1}} \times \sum_{C \in S_1} \sum_{\tilde{C} \in S_2} \left[1 - \frac{2}{\pi} \tan^{-1} \sqrt{\frac{t}{n - t}} \right] \\
&= 1 - \frac{1}{\binom{n-1}{k} \times \binom{n-1}{k-1}} \times \sum_{C \in S_1} \sum_{\tilde{C} \in S_2} \left[\frac{2}{\pi} \tan^{-1} \sqrt{\frac{t}{n - t}} \right].
\end{aligned}$$

We further simplify this. For every pair of inputs $C \in S_1$ and $\tilde{C} \in S_2$, one can compute $(1 - \frac{2}{\pi} \tan^{-1} \sqrt{\frac{t}{n-t}})$ as follows. Note that for every value of t , $\tan^{-1} \sqrt{\frac{t}{n-t}}$ and $\tan^{-1} \sqrt{\frac{n-t}{t}}$ both term will occur in the summation $\sum_{C \in S_1} \sum_{\tilde{C} \in S_2} \frac{2}{\pi} \tan^{-1} \sqrt{\frac{t}{n-t}}$. That means the above summation will contain $\tan^{-1} \sqrt{x} + \tan^{-1} \sqrt{\frac{1}{x}} = \frac{\pi}{2}$, for different values of x . As there are total $\binom{n-1}{k} \times \binom{n-1}{k-1}$ terms in the summation, the final expectation of $\mathcal{A}_f^{E_{n,k}}$ will be equal to $\frac{1}{2}$ for $f \in \mathcal{B}_n^{\text{PUF}}$. This completes our proof. \square

Let us provide an example with $n = 9$ and $k = 4$, i.e., $|E_{n,k}| = 126$. Hence $|S_1| = \binom{8}{3} = 56$ and $|S_2| = \binom{8}{4} = 70$. Let $T_i = \{(C, \tilde{C}) \in S_1 \times S_2 : t = i\}$. It can be checked that $|T_i| = |T_{n-i}|$, for $i = 1, \dots, 8$. In $E_{9,4}$, $|T_1| = |T_8| = 35$, i.e., there are 35 pair of inputs $(C, \tilde{C}) \in S_1 \times S_2$, for which $t = 1$ and another different 35 pairs of inputs $(C, \tilde{C}) \in S_1 \times S_2$, for which $t = 8$. If we add $(1 - \frac{2}{\pi} \tan^{-1} \sqrt{\frac{t}{n-t}})$ for all these 70 pairs of distinct inputs, the final value becomes 35. Similarly $|T_2| = |T_7| = 215$, $|T_3| = |T_6| = 635$ and $|T_4| = |T_5| = 1075$. Hence the final expectation becomes $\frac{1}{8C_3 \times 8C_4} \times [35 + 215 + 635 + 1075] = \frac{1}{2}$.

From the result of Theorem 4 it can be observed that if the challenge pairs are chosen with certain restrictions related to the input weights, then there does not exist any bias in the output of the Arbiter PUF. Thus in such restricted scenarios, such simple models of physically unclonable devices might provide acceptable cryptographic parameters.

In a related note, it has been shown [MMM⁺19] that certain cryptographic properties related to the Walsh spectrum of a Boolean function degrades in the restricted domain. Here we show that in the case of Arbiter PUFs, certain kind of autocorrelation property in a restricted sense, improves. The proposed notion of restricted autocorrelation might be explored for analyzing the security of FLIP [MJSC16] type ciphers under differential attack or related key attack.

3.6 Conclusion

In this chapter, we have studied certain limitations of Arbiter PUFs and shown that the class of Boolean functions constructed using n -length ($n > 1$) PUFs is a proper subset of the set of all n variable Boolean functions. We make some combinatorial analysis of n -length Arbiter PUFs. We explore efficient techniques towards distin-

guishing whether a Boolean function truth table can indeed be fabricated through an Arbiter PUF or not. Further we have looked at autocorrelation in certain restricted sense and presented relevant results in this direction.

On Combining Arbiter-based PUFs

Contents

4.1 Introduction	78
4.2 Understanding the relation between $\mathcal{B}_n^{\text{XOR-PUF}}$, $\mathcal{B}_{n,m}^{\text{XOR-PUF}}$ and \mathcal{B}_n	79
4.3 Theoretical estimation of bias in PUF with combiner function	86
4.4 Conclusion	89

In Chapter 3, it was shown that a negligible portion of Boolean functions class can be generated from an n -length Arbiter-based PUF. While studying this independently, we missed the link between Arbiter PUF and threshold functions [Muroga et al., 1961]. In this chapter, we consider this link and first point out some miscalculations done by Yajima and Ibaraki in 1965, regarding the number of Boolean functions generated from a threshold function. Then we study the combination (extendability) of Arbiter PUFs and investigate the XOR of two n -length such devices with arbitrary inputs. We checked, for some small dimensions, that among all the XORNF (Exclusive OR Normal Form) representations of two Arbiter PUFs of the same length, the XOR of two Arbiter PUFs of the same length produces a large portion of the Boolean functions class. Based upon extensive computations, we therefore propose a conjecture. Finally, we concentrate on a general PUF combiner, and compute the probability of two outputs from a combiner model PUF corresponding to two different challenge inputs being the same.

4.1 Introduction

In 1961, the Majority Decision Function or Threshold Function was introduced and investigated by Muroga, Toda and Takasu [MTT61]. The exact number of n -variable threshold functions is exactly the same as the number of Boolean functions generated from n -length Arbiter PUFs. In [Sor17, Section 6.1.2], the author noted that an Arbiter PUF can be modeled as a threshold function. Recently, we were also studying such functions independently and unfortunately were not aware of [MTT61, Sor17, YI65] and thus the connection with threshold functions was not referred to in [RRM21, SBC⁺19b]. Here, we point out some miscalculations done by Yajima and Ibaraki in [YI65], regarding the number of Boolean functions generated from a threshold function.

In this chapter we show how one can explore a significant portion of the Boolean functions class from the XOR of two n -length PUFs with the same inputs. The number of n -bit Boolean functions that can be generated from the circuit of an n -length Arbiter PUF was recently studied in [RRM21]. It has been shown in [RRM21] that a very small proportion of the complete set of Boolean functions can be generated using Arbiter PUFs. This motivates us to study the number of Boolean functions that can be generated from the circuit of an XOR-PUF [RSS⁺10b]. Here, we consider two different versions of XOR-PUFs, one having the same input lengths and the other having different input lengths. We also consider the XORNF (Exclusive OR Normal Form) representation of two Arbiter PUFs of the same length and noted the number of Boolean functions generated from these.

It has been shown in [SBC⁺19b] that if one considers two (challenge) inputs from $\{-1, 1\}^n$, $C = (C_1, C_2, \dots, C_n)$ and $\tilde{C} = (\tilde{C}_1, C_2, \dots, C_n)$ with the condition $C_1 + \tilde{C}_1 = 0$, then the output from PUF corresponding to C and \tilde{C} matches with high probability. In the same paper the authors have provided a generic form of such kind of probabilities for different parameters. Such kind of probability for PUF with combiner function for a special case is also derived in [SBC⁺19b], although the more general case was not addressed in [SBC⁺19b]. In this direction, we derive the general form of the probability in which two outputs from the PUF with combiner model will be equal corresponding to two different challenge inputs.

Remark 1. *An n -length Arbiter PUF can be modeled as a threshold function with*

weight vector $W = (\alpha_1, \alpha_2 + \beta_1, \alpha_3 + \beta_2, \dots, \beta_n)$ and threshold 0 (see [Sor17, Lemma 6.2]). In this direction, some analysis was done in [MTT61, YI65]. In [YI65], it is mentioned that $f = f_1 \parallel f_2$ is an $(n+1)$ -variable threshold function if and only if both f_1 and f_2 are n -variable threshold functions with respect to the same weight vector W . From the definition of a PUF one can easily check that the threshold of each PUF is always 0. So, if both f_1 and f_2 are n -variable threshold functions with respect to the same weight vector $W = (\alpha_1, \alpha_2 + \beta_1, \alpha_3 + \beta_2, \dots, \beta_n)$, then f_1 and f_2 are identical. Hence, according to a result of [YI65], $f_1 \parallel f_1$ becomes a threshold function. Since a PUF can be modeled as a threshold function, then $f_1 \parallel f_1 \in \mathcal{B}_{n+1}^{\text{PUF}}$, which contradicts Proposition 7.

4.2 Understanding the relation between $\mathcal{B}_n^{\text{XOR-PUF}}$, $\mathcal{B}_{n,m}^{\text{XOR-PUF}}$ and \mathcal{B}_n

An n -length Arbiter PUF can not generate all possible functions in n -variables. In [RRM21] it was shown that a very small portion of the complete set of Boolean functions \mathcal{B}_n can be generated using Arbiter PUFs. In this context, we now present some combinatorial results on the number of Boolean functions that can be generated from the XOR of two Arbiter PUFs.

4.2.1 The relation between $\mathcal{B}_{n,m}^{\text{XOR-PUF}}$ and $\mathcal{B}_n^{\text{PUF}}$

We now explore the class $\mathcal{B}_{n,m}^{\text{XOR-PUF}}$ of $(n+m)$ -variable Boolean functions generated from the XOR of two Arbiter PUFs of length n and m , respectively. We first perform experiments to measure $|\mathcal{B}_{n,m}^{\text{XOR-PUF}}|$ for different values of (n, m) and we observe that $|\mathcal{B}_{n,m}^{\text{XOR-PUF}}| = 8, 28, 98, 208$ for $(n, m) = (1, 1), (2, 1), (2, 2), (3, 1)$ respectively. From these experiments on $|\mathcal{B}_{n,m}^{\text{XOR-PUF}}|$ we observe that $|\mathcal{B}_{n,1}^{\text{XOR-PUF}}| = 2|\mathcal{B}_n^{\text{PUF}}|$, for $n = 1, 2, 3$. It turns out that our observation is true in general, namely, we prove the following theorem.

Theorem 5. For any $n, m \in \mathbb{N}$, $|\mathcal{B}_{n,m}^{\text{XOR-PUF}}| = \frac{1}{2}|\mathcal{B}_n^{\text{PUF}}| \cdot |\mathcal{B}_m^{\text{PUF}}|$.

Proof. Let $f_1 \in \mathcal{B}_n^{\text{PUF}}$ and $f_2 \in \mathcal{B}_m^{\text{PUF}}$, where $n, m \in \mathbb{N}$. Without loss of generality we can assume that $n \geq m$. We consider the Boolean function $F = f_1 \oplus f_2 \in \mathcal{B}_{n,m}^{\text{XOR-PUF}}$.

Here we have $|\mathcal{B}_n^{\text{PUF}}|$ choices for f_1 and $|\mathcal{B}_m^{\text{PUF}}|$ choices for f_2 . From the result of Roy et al. [RRM21] we know that f_2 also belongs to $\mathcal{B}_n^{\text{PUF}}$, as $n \geq m$. Thus we infer that $(f_1 \oplus f_2)$ belongs to $\mathcal{B}_n^{\text{XOR-PUF}}$. We also know that the zero function always belongs to any $\mathcal{B}_m^{\text{PUF}}$. Using these two facts we find that $F = f_1 \oplus f_2 = ((f_2 \oplus f_1) \oplus \mathbf{0}) \in \mathcal{B}_{n,m}^{\text{XOR-PUF}}$. Hence $|\mathcal{B}_{n,m}^{\text{XOR-PUF}}| = \frac{1}{2}|\mathcal{B}_n^{\text{PUF}}| \cdot |\mathcal{B}_m^{\text{PUF}}|$. \square

In [RRM21] Roy et al. proved that if a Boolean function f belongs to $\mathcal{B}_n^{\text{PUF}}$ then $(1 \oplus f)$ also belongs to $\mathcal{B}_n^{\text{PUF}}$. In the same direction we state and prove the following lemma.

Lemma 2. *For any Boolean function $f \in \mathcal{B}_{n,m}^{\text{XOR-PUF}}$ if and only if $(1 \oplus f) \in \mathcal{B}_{n,m}^{\text{XOR-PUF}}$.*

Proof. We first assume that $f \in \mathcal{B}_{n,m}^{\text{XOR-PUF}}$. That is, there exists $f_1 \in \mathcal{B}_n^{\text{PUF}}$ and $f_2 \in \mathcal{B}_m^{\text{PUF}}$ for which $f = f_1 \oplus f_2$. Now we consider the function $(1 \oplus f) = (1 \oplus f_1 \oplus f_2) = ((1 \oplus f_1) \oplus f_2)$. From the result of Roy et al. [RRM21] we get that $(1 \oplus f_1)$ will also belong to $\mathcal{B}_n^{\text{PUF}}$. Hence, $(1 \oplus f) \in \mathcal{B}_{n,m}^{\text{XOR-PUF}}$ as $(1 \oplus f_1) \in \mathcal{B}_n^{\text{PUF}}$ and $f_2 \in \mathcal{B}_m^{\text{PUF}}$. The other direction of the lemma follows similarly. \square

Lemma 3. *If $f_1 \notin \mathcal{B}_{n,m}^{\text{XOR-PUF}}$ then $f_1 \parallel f$ or $f \parallel f_1 \notin \mathcal{B}_{n+1,m}^{\text{XOR-PUF}}$, for any $f \in \mathcal{B}_{n+m}$.*

Proof. We assume there exists an $(n+1+m)$ -variable Boolean function $F = f_1 \parallel f \in \mathcal{B}_{n+1,m}^{\text{XOR-PUF}}$ such that $f_1 \notin \mathcal{B}_{n,m}^{\text{XOR-PUF}}$ for some $f \in \mathcal{B}_{n+m}$. Let the $(n+1+m)$ -bit challenge input of $\Delta_{n+1,m}^{\text{XOR}}(C)$ be $C = (C_1, \dots, C_{n+1}, \dots, C_{(n+1)+m})$. The mathematical model of an $(n+1+m)$ -length XOR-PUF constructed from two Arbiter PUFs of length $(n+1)$ and m is $\Delta_{n+1,m}^{\text{XOR}}(C) = \Delta(C_1, C_2, \dots, C_{n+1}) \cdot \Delta'(C_{(n+1)+1}, \dots, C_{(n+1)+m})$, where

$$\Delta(C_1, C_2, \dots, C_{n+1}) = \alpha_1 P_0 + (\alpha_2 + \beta_1) P_1 + \dots + (\alpha_{n+1} + \beta_n) P_n + \beta_{n+1} \quad (4.1)$$

$$\Delta'(C_{(n+1)+1}, \dots, C_{(n+1)+m}) = \alpha'_1 P'_0 + (\alpha'_2 + \beta'_1) P'_1 + \dots + (\alpha'_m + \beta'_{m-1}) P'_{m-1} + \beta'_m. \quad (4.2)$$

Since $F \in \mathcal{B}_{n+1,m}^{\text{XOR-PUF}}$, the inequalities constructed from the expression of $\Delta_{n+1,m}^{\text{XOR}}(C)$ and the truth table corresponding to F will provide a solution for $\alpha_i, \beta_i, \alpha'_i$ and β'_i . Now in the truth table of F , the upper half is for $C_{n+1} = 1$, which is exactly

same as the truth table of f_1 and the lower half is for $C_{n+1} = -1$, which is exactly same as the truth table of f . Now we construct an n -length Arbiter PUF $\Delta''(C_1, C_2, \dots, C_n)$ from Equation (4.1) by substituting $\alpha_i'' = \alpha_i$ for $i = 1, 2, \dots, n$; $\beta_i'' = \beta_i$ for $i = 1, 2, \dots, n-1$ and $\beta_n'' = (\alpha_{n+1} + \beta_n) + \beta_{n+1}$. Then the existence of α_i and β_i guarantees that the $(n+m)$ -length XOR-PUF constructed from $\Delta''(C_1, C_2, \dots, C_n)$ and $\Delta'(C_{(n+1)+1}, \dots, C_{(n+1)+m})$ will be able to generate the truth table of f_1 . Hence $f_1 \in \mathcal{B}_{n,m}^{\text{XOR-PUF}}$. This is a contradiction. Hence $F = f_1 \parallel f \notin \mathcal{B}_{n+1,m}^{\text{XOR-PUF}}$. Similarly we can prove $f \parallel f_1 \notin \mathcal{B}_{n+1,m}^{\text{XOR-PUF}}$. \square

4.2.2 The relation between $\mathcal{B}_n^{\text{XOR-PUF}}$ and $\mathcal{B}_n^{\text{PUF}}$

In this section, we explore the class of Boolean functions generated from the XOR of two n -length Arbiter PUFs, i.e., $\mathcal{B}_n^{\text{XOR-PUF}}$. We start with $n = 1$. We prove the following proposition to explore the number of Boolean functions that can be generated from the XOR of two n -length PUFs.

Proposition 8. *All possible Boolean functions involving one and two variables can be generated by using the XOR of two Arbiter PUFs of length 1 and 2, respectively, i.e., $\mathcal{B}_1^{\text{XOR-PUF}} = \mathcal{B}_1$ and $\mathcal{B}_2^{\text{XOR-PUF}} = \mathcal{B}_2$.*

Proof. This proposition was shown by exhaustively enumerating $\mathcal{B}_1^{\text{XOR-PUF}}$ and $\mathcal{B}_2^{\text{XOR-PUF}}$. We considered 1 and 2-length XOR-PUFs for different random delay parameters and observed that all the truth tables are generated in our experiment. \square

We now consider the case for $n = 3$. There are total $2^{2^3} = 256$ many Boolean functions in this case. We observed from our experiments that $|\mathcal{B}_3^{\text{XOR-PUF}}| = 254$. In this regard, we state the following result.

Proposition 9. *Two 3-variable Boolean functions $f_1 = (0\ 1\ 0\ 1\ 1\ 0\ 1\ 0)$, $f_2 = (1\ 0\ 1\ 0\ 0\ 1\ 0\ 1)$ do not belong to $\mathcal{B}_3^{\text{XOR-PUF}}$.*

Proof. The mathematical model of the XOR of two 3-length PUFs is $\Delta^{\text{XOR}}(C) = \Delta(C) \cdot \Delta'(C)$, where $\Delta(C) = (\alpha_1 P_0 + (\alpha_2 + \beta_1) P_1 + (\alpha_3 + \beta_2) P_2 + \beta_3)$ and $\Delta'(C) = (\alpha'_1 P_0 + (\alpha'_2 + \beta'_1) P_1 + (\alpha'_3 + \beta'_2) P_2 + \beta'_3)$. The final truth table of f_1 will be of the following form:

C_3	C_2	C_1	$\Delta^{XOR}(C)$	f_1
1	1	1	$\Delta(1, 1, 1) \cdot \Delta'(1, 1, 1) < 0$	0
1	1	-1	$\Delta(1, 1, -1) \cdot \Delta'(1, 1, -1) > 0$	1
1	-1	1	$\Delta(1, -1, 1) \cdot \Delta'(1, -1, 1) < 0$	0
1	-1	-1	$\Delta(1, -1, -1) \cdot \Delta'(1, -1, -1) > 0$	1
-1	1	1	$\Delta(-1, 1, 1) \cdot \Delta'(-1, 1, 1) > 0$	1
-1	1	-1	$\Delta(-1, 1, -1) \cdot \Delta'(-1, 1, -1) < 0$	0
-1	-1	1	$\Delta(-1, -1, 1) \cdot \Delta'(-1, -1, 1) > 0$	1
-1	-1	-1	$\Delta(-1, -1, -1) \cdot \Delta'(-1, -1, -1) < 0$	0

Let $K = \alpha_1 \cdot \alpha'_1 + (\alpha_2 + \beta_1) \cdot (\alpha'_2 + \beta'_1) + (\alpha_3 + \beta_2) \cdot (\alpha'_3 + \beta'_2) + \beta_3 \cdot \beta'_3$. If we add four > 0 inequalities we will get $K > 0$ and adding four < 0 inequalities we will get $K < 0$, which generates a contradiction. Hence $f_1 \notin \mathcal{B}_3^{XOR-PUF}$. Similarly it can be shown that $f_2 \notin \mathcal{B}_3^{XOR-PUF}$. \square

Proposition 9 along with our experiments justify that $|\mathcal{B}_3^{XOR-PUF}| = 254$. Also observe that the two Boolean functions $f_2 = f_1 \oplus 1$. This observation suggests the following result.

Lemma 4. *For any n -variable Boolean function $f \in \mathcal{B}_n^{XOR-PUF}$ if and only if $(1 \oplus f) \in \mathcal{B}_n^{XOR-PUF}$.*

Proof. We first assume that $f \in \mathcal{B}_n^{XOR-PUF}$. For this Boolean function f there will exist $f_1, f_2 \in \mathcal{B}_n^{PUF}$ such that $f = f_1 \oplus f_2$. Now we consider $(1 \oplus f) = (1 \oplus f_1 \oplus f_2) = (1 \oplus f_1) \oplus f_2$. From the result of Roy et al. [RRM21] it follows that $(1 \oplus f_1)$ also belongs to \mathcal{B}_n^{PUF} . Hence we can directly say that $((1 \oplus f_1) \oplus f_2) \in \mathcal{B}_n^{XOR-PUF}$. The opposite direction of the theorem follows similarly. \square

Lemma 5. *If $f_1 \notin \mathcal{B}_n^{XOR-PUF}$, then there does not exist any $F \in \mathcal{B}_{n+1}^{XOR-PUF}$ where F is of the form $f_1 \parallel f$ or $f \parallel f_1$, regardless of $f \in \mathcal{B}_n$.*

Proof. We first assume that there exists an $(n + 1)$ -variable Boolean function $F = f_1 \parallel f \in \mathcal{B}_{n+1}^{XOR-PUF}$, where $f_1 \notin \mathcal{B}_n^{XOR-PUF}$. The mathematical model of an $(n + 1)$ -length XOR-PUF constructed from two $(n + 1)$ -length Arbiter PUF, $\Delta(C)$ and $\Delta'(C)$ is given by $\Delta^{XOR}(C) = \Delta(C) \cdot \Delta'(C)$, where $C = (C_1, C_2, \dots, C_{n+1})$ and

$$\Delta(C_1, C_2, \dots, C_{n+1}) = \alpha_1 P_0 + (\alpha_2 + \beta_1) P_1 + \dots + (\alpha_{n+1} + \beta_n) P_n + \beta_{n+1} \quad (4.3)$$

$$\Delta'(C_1, C_2, \dots, C_{n+1}) = \alpha'_1 P_0 + (\alpha'_2 + \beta'_1) P_1 + \dots + (\alpha'_{n+1} + \beta'_n) P_n + \beta'_{n+1}. \quad (4.4)$$

Since $F \in \mathcal{B}_{n+1}^{\text{XOR-PUF}}$, the inequalities constructed from $\Delta^{\text{XOR}}(C)$ and the truth table of F will give a solution for $\alpha_i, \beta_i, \alpha'_i$ and β'_i for $i \in \{1, 2, \dots, n+1\}$. The upper half of the truth table of F corresponding to $C_{n+1} = 1$, which is same as the truth table of f_1 and the lower half of the truth table of F corresponding to $C_{n+1} = -1$, which is same as the truth table of f . Now, we assume an n -length XOR-PUF, constructed from two n -length Arbiter PUFs of delay differences $\Delta_1(C')$ and $\Delta_2(C')$, given by $\Delta^{\text{XOR}}(C') = \Delta_1(C') \cdot \Delta_2(C')$, where $C' = (C_1, C_2, \dots, C_n)$, $\alpha_{1i} = \alpha_i, \alpha_{2i} = \alpha'_i$ for $i = 1, 2, \dots, n$, $\beta_{1i} = \beta_i, \beta_{2i} = \beta'_i$ for $i = 1, 2, \dots, n-1$, $\beta_{1n} = (\alpha_{n+1} + \beta_n) P_n + \beta_{n+1}$, $\beta_{2n} = (\alpha'_{n+1} + \beta'_n) P_n + \beta'_{n+1}$ and

$$\Delta_1(C') = \alpha_{11} P_0 + (\alpha_{12} + \beta_{11}) P_1 + \dots + (\alpha_{1n} + \beta_{1(n-1)}) P_{n-1} + \beta_{1n} \quad (4.5)$$

$$\Delta_2(C') = \alpha_{21} P_0 + (\alpha_{22} + \beta_{21}) P_1 + \dots + (\alpha_{2n} + \beta_{2(n-1)}) P_{n-1} + \beta_{2n}. \quad (4.6)$$

Then the existence of $\alpha_i, \beta_i, \alpha'_i$ and β'_i guarantees that the n -length XOR-PUF constructed from two n -length Arbiter PUFs, of delay differences $\Delta_1(C')$ and $\Delta_2(C')$ will be able to generate the truth table of f_1 . Hence $f_1 \in \mathcal{B}_n^{\text{XOR-PUF}}$, which is a contradiction. Hence $F = f_1 \parallel f \notin \mathcal{B}_{n+1}^{\text{XOR-PUF}}$. Similarly we can prove that $f \parallel f_1 \notin \mathcal{B}_{n+1}^{\text{XOR-PUF}}$. \square

In [RRM21], Roy et al. proved that $f_1 = (1\ 1\ 0\ 1\ 1\ 0\ 1\ 0)$, $f_2 = (1\ 1\ 1\ 0\ 0\ 1\ 0\ 1) \notin \mathcal{B}_3^{\text{PUF}}$ but here we notice that $f_1 \oplus f_2 = (0\ 0\ 1\ 1\ 1\ 1\ 1\ 1) \in \mathcal{B}_3^{\text{XOR-PUF}}$. The function $f_1 \oplus f_2$ belongs to $\mathcal{B}_3^{\text{XOR-PUF}}$ due to the fact that it can also be expressed as the XOR of two functions f_3, f_4 from $\mathcal{B}_3^{\text{PUF}}$ (for example $f_3 = (0\ 0\ 0\ 0\ 0\ 0\ 0\ 0)$ and $f_4 = (0\ 0\ 1\ 1\ 1\ 1\ 1\ 1)$). Moreover, $f_1 = (1\ 1\ 0\ 1\ 1\ 0\ 1\ 0)$, $f_2 = (1\ 1\ 1\ 0\ 0\ 1\ 0\ 1) \in \mathcal{B}_3^{\text{XOR-PUF}}$ as both f_1, f_2 can be expressed as the XOR of two functions from $\mathcal{B}_3^{\text{PUF}}$. With this we summarize the values of $|\mathcal{B}_n^{\text{XOR-PUF}}|$ vis-a-vis $|\mathcal{B}_n^{\text{PUF}}|$ in Table 4.1.

From Lemma 5 we can directly say that if $f \in \mathcal{B}_{n+1}^{\text{XOR-PUF}}$, then $f = f_1 \parallel f_2$, where $f_1, f_2 \in \mathcal{B}_n^{\text{XOR-PUF}}$. This observation suggests the following result.

Lemma 6. *If $f \in \mathcal{B}_n^{\text{XOR-PUF}}$, then $f \parallel f \in \mathcal{B}_{n+1}^{\text{XOR-PUF}}$.*

Proof. Let $F = f \parallel f$. Since $f \in \mathcal{B}_n^{\text{XOR-PUF}}$, the mathematical model of f constructed

n	$ \mathcal{B}_n^{\text{PUF}} $	$ \mathcal{B}_n^{\text{XOR-PUF}} $
1	4	4
2	14	16
3	104	254
4	1882	54310

Table 4.1: $|\mathcal{B}_n^{\text{XOR-PUF}}|$ for different n

from two n -length Arbiter PUF, $\Delta(C)$ and $\Delta'(C)$ is given by $\Delta^{\text{XOR}}(C) = \Delta(C) \cdot \Delta'(C)$, where $C = (C_1, C_2, \dots, C_{n+1})$ and

$$\Delta(C_1, C_2, \dots, C_n) = \alpha_1 P_0 + (\alpha_2 + \beta_1) P_1 + \dots + (\alpha_n + \beta_{n-1}) P_{n-1} + \beta_n \quad (4.7)$$

$$\Delta'(C_1, C_2, \dots, C_n) = \alpha'_1 P_0 + (\alpha'_2 + \beta'_1) P_1 + \dots + (\alpha'_n + \beta'_{n-1}) P_{n-1} + \beta'_n. \quad (4.8)$$

Now we consider the mathematical model of an $(n + 1)$ -length XOR-PUF and the implied inequalities using the truth values of $F = f \parallel f$. Let the delay difference of the $(n + 1)$ -length XOR-PUF is given by $\Delta^{\text{XOR}}(C') = \Delta_1(C') \cdot \Delta_2(C')$, where $C' = (C_1, C_2, \dots, C_{n+1})$ and

$$\Delta_1(C') = \alpha_{11} P_0 + (\alpha_{12} + \beta_{11}) P_1 + \dots + (\alpha_{1(n+1)} + \beta_{1n}) P_n + \beta_{1(n+1)} \quad (4.9)$$

$$\Delta_2(C') = \alpha_{21} P_0 + (\alpha_{22} + \beta_{21}) P_1 + \dots + (\alpha_{2(n+1)} + \beta_{2n}) P_n + \beta_{2(n+1)} \quad (4.10)$$

We break the truth table of F in two parts. In the top part we take $C_{n+1} = 1$ and the bottom part we take $C_{n+1} = -1$. The form of each row corresponding to $C_{n+1} = 1$ is of the form

$$(\alpha_{11} P_0 + (\alpha_{12} + \beta_{11}) P_1 + \dots + (\alpha_{1(n+1)} + \beta_{1n}) P_n + \beta_{1(n+1)}) \cdot (\alpha_{21} P_0 + (\alpha_{22} + \beta_{21}) P_1 + \dots + (\alpha_{2(n+1)} + \beta_{2n}) P_n + \beta_{2(n+1)}) \quad (4.11)$$

Similarly each row corresponding to $C_{n+1} = -1$ is of the form

$$(-\alpha_{11} P_0 - (\alpha_{12} + \beta_{11}) P_1 - \dots - (\alpha_{1(n+1)} + \beta_{1n}) P_n - \beta_{1(n+1)}) \cdot (-\alpha_{21} P_0 - (\alpha_{22} + \beta_{21}) P_1 - \dots - (\alpha_{2(n+1)} + \beta_{2n}) P_n - \beta_{2(n+1)})$$

$$\cdots - (\alpha_{2(n+1)} + \beta_{2n})P_n + \beta_{2(n+1)} \quad (4.12)$$

The inequalities sign will be decided by the truth values of $F = f \parallel f$. Now if we choose $\alpha_{1(n+1)} = \beta_{1(n+1)} = 0 = \alpha_{2(n+1)} = \beta_{2(n+1)}$, then the system of inequalities corresponding to $C_{n+1} = 1$ will be identical with the system of inequalities corresponding to $C_{n+1} = -1$. Since $f \in \mathcal{B}_n^{\text{XOR-PUF}}$, one can obtain a solution from the two system of inequalities whose signs are decided by the truth values of f . The existence of solution proves that $F = f \parallel f \in \mathcal{B}_{n+1}^{\text{XOR-PUF}}$. \square

From Lemma 5 and Lemma 6, the following result follows.

Corollary 1. *For any value of n , $|\mathcal{B}_n^{\text{XOR-PUF}}| \leq |\mathcal{B}_{n+1}^{\text{XOR-PUF}}| \leq |\mathcal{B}_n^{\text{XOR-PUF}}|^2$.*

Thus it is clearly understood that a larger class of Boolean functions can be generated from the XOR of two n -length Arbiter PUFs with the same input. Moreover among all the XORNF (Exclusive OR Normal Form) representations of two n -length Arbiter PUFs, this variant only can generate a significant number of Boolean functions. To check this, we first consider two n -length Arbiter PUFs f_1 and f_2 . Let $g(d_0, d_1, d_2, d_3) = d_0 \oplus (d_1 \cdot f_1) \oplus (d_2 \cdot f_2) \oplus (d_3 \cdot f_1 \cdot f_2)$, where $d_0, d_1, d_2, d_3 \in \{0, 1\}$. Let $d = (d_0, d_1, d_2, d_3)$ and let us define $\mathcal{B}_n^{\mathbf{g}(d)}$ to denote the set of all n -variable Boolean functions generated from the function $g(d)$. For example, if $d_0 = 0, d_1 = 1, d_2 = 1, d_3 = 0$, then $g(0, 1, 1, 0) = f_1 \oplus f_2$ and $\mathcal{B}_n^{\mathbf{g}(d)} = \mathcal{B}_n^{\text{XOR-PUF}}$. The exact count of $|\mathcal{B}_n^{\mathbf{g}(d)}|$ for each of the 16 values of d are mentioned in Table 4.2.

Observation 1. *From the result of Proposition 9, we have $\mathcal{B}_n^{\text{XOR-PUF}} \subset \mathcal{B}_n$, for $n \geq 3$. However, if we add one more Arbiter PUF and take the XOR of the outputs produced by the 3 PUFs, then we observe experimentally that $\mathcal{B}_3^{3\text{-XOR-PUF}} = \mathcal{B}_3$, where $\mathcal{B}_n^{3\text{-XOR-PUF}}$ is the set of all n -variable Boolean functions generated from the XOR of three n -length Arbiter PUFs. We also observed that $|\mathcal{B}_4^{3\text{-XOR-PUF}}| = 2^{2^4} - 2$. But the theoretical proof for 3 XOR-PUF becomes more complex. We have performed experiments and found two functions $F_1, F_2 \notin \mathcal{B}_4^{3\text{-XOR-PUF}}$, where $F_1 = f_1 \parallel f_1$ and $F_2 = f_2 \parallel f_2$ such that $f_1, f_2 \notin \mathcal{B}_3^{\text{XOR-PUF}}$.*

Based upon extensive experiments, we propose the following conjecture.

Conjecture 1. *Let $n \geq 3$ be an integer. Then $\mathcal{B}_n^{(n-1)\text{-XOR-PUF}} \subset \mathcal{B}_n$, where $\mathcal{B}_n^{\mathbf{k}\text{-XOR-PUF}}$ is the set of all n -variable Boolean functions generated from the XOR of k many n -length Arbiter PUFs.*

$d = (d_0, d_1, d_2, d_3)$	$g(d)$	$ \mathcal{B}_1^{g(d)} $	$ \mathcal{B}_2^{g(d)} $	$ \mathcal{B}_3^{g(d)} $	$ \mathcal{B}_4^{g(d)} $
(0, 0, 0, 0)	0	1	1	1	1
(0, 0, 0, 1)	$f_1 \cdot f_2$	4	16	246	32286
(0, 0, 1, 0)	f_2	4	14	104	1882
(0, 0, 1, 1)	$f_2 \oplus f_1 \cdot f_2$	4	16	246	32286
(0, 1, 0, 0)	f_1	4	14	104	1882
(0, 1, 0, 1)	$f_1 \oplus f_1 \cdot f_2$	4	16	246	32286
(0, 1, 1, 0)	$f_1 \oplus f_2$	4	16	254	54310
(0, 1, 1, 1)	$f_1 \oplus f_2 \oplus f_1 \cdot f_2$	4	16	246	32286
(1, 0, 0, 0)	1	1	1	1	1
(1, 0, 0, 1)	$1 \oplus f_1 \cdot f_2$	4	16	246	32286
(1, 0, 1, 0)	$1 \oplus f_2$	4	14	104	1882
(1, 0, 1, 1)	$1 \oplus f_2 \oplus f_1 \cdot f_2$	4	16	246	32286
(1, 1, 0, 0)	$1 \oplus f_1$	4	14	104	1882
(1, 1, 0, 1)	$1 \oplus f_1 \oplus f_1 \cdot f_2$	4	16	246	32286
(1, 1, 1, 0)	$1 \oplus f_1 \oplus f_2$	4	16	254	54310
(1, 1, 1, 1)	$1 \oplus f_1 \oplus f_2 \oplus f_1 \cdot f_2$	4	16	246	32286

Table 4.2: $|\mathcal{B}_n^{g(d)}|$ for different $n \leq 4$

4.3 Theoretical estimation of bias in PUF with combiner function

In literature different kind of analysis on estimation of non-randomness in different models of PUFs have been performed. In this section we provide a generic expression of the probability of final output from the PUFs with combiner model to be equal for different challenge inputs to the PUFs. We derive the expression of that probability in Theorem 7. It has been shown in [SBC⁺19b] that if one considers two (challenge) inputs from $\{-1, 1\}^n$, $C = (C_1, C_2, \dots, C_n)$ and $\tilde{C} = (\tilde{C}_1, C_2, \dots, C_n)$ with the condition $C_1 + \tilde{C}_1 = 0$, then the output from PUF corresponding to C and \tilde{C} matches with high probability. In the same paper the authors have provided a generic form of these probabilities for different parameters, as well as for a PUF via a combiner function for a special case, although the more general case was not addressed in [SBC⁺19b]. Similar to these efforts there are some statistical and machine learning techniques that present non-randomness observations. Here we present a generalized result in this regard. We derive the expression of that probability in Theorem 7.

Before proceeding to Theorem 7 we state a theorem from [SBC⁺19b] which will

be used in the proof of our new result.

Theorem 6 (Siddhanti et al. [SBC⁺19b]). *Let $f : \{0, 1\}^n \rightarrow \{0, 1\}$ be a Boolean function. We partition the autocorrelation spectrum with respect to the weight of the vectors, and define the set $\Omega_f^{(-2^n+2k, w)} = \{a : \mathcal{C}_f(a) = -2^n + 2k, wt(a) = w\}$. Further, we assume that the probability of any two input changes is $Pr[x_i + \tilde{x}_i = 0] = p$, for all i , where $\tilde{x}_i = x_i + a_i$, for $a_i \in \{0, 1\}$. Then,*

$$Pr[f(x_1, \dots, x_n) = f(\tilde{x}_1, \dots, \tilde{x}_n)] = \frac{1}{2^n} \sum_{k=1}^{2^{n+1}} \sum_{w=0}^n kp^{n-w}(1-p)^w \left| \Omega_f^{(-2^n+2k, w)} \right|.$$

Theorem 7. *Let $f : \{0, 1\}^n \rightarrow \{0, 1\}$ be a Boolean function. We let f_0, f_1 be the first and the second half of the function (where the input space is ordered lexicographically), and therefore, $f(x_1, \dots, x_n) = \bar{x}_n f_0(x_1 \dots, x_{n-1}) + x_n f_1(x_1 \dots, x_{n-1})$. Let $\Omega_{f_i}^{(-2^{n-1}+2k, w)} = \{a : \mathcal{C}_{f_i}(a) = -2^{n-1} + 2k, wt(a) = w\}$, $i = 0, 1$. Further, we assume that the probability of any two input changes is $Pr[x_i + \tilde{x}_i = 0] = p$, for all $1 \leq i \leq n - \ell$, and $Pr[x_j + \tilde{x}_j = 0] = p_j$, $n - \ell + 1 \leq j \leq n$, where $\tilde{x}_k = x_k + a_k$, for $a_k \in \{0, 1\}$. Then,*

$$\begin{aligned} & Pr[f(x_1, \dots, x_n) = f(\tilde{x}_1, \dots, \tilde{x}_n)] \\ &= \frac{\prod_{j=0}^{\ell-1} p_{n-j}}{2^{\frac{\ell(2n-\ell-1)}{2}}} \sum_{k=1}^{2^{n-\ell}} \sum_{w=0}^{n-\ell} \left(\left| \Omega_{f_{0\dots 00}}^{(-2^{n-\ell}+2k, w)} \right| + \left| \Omega_{f_{0\dots 01}}^{(-2^{n-\ell}+2k, w)} \right| + \dots + \left| \Omega_{f_{1\dots 11}}^{(-2^{n-\ell}+2k, w)} \right| \right). \end{aligned}$$

In particular, if $\ell = 1$,

$$\begin{aligned} & Pr[f(x_1, \dots, x_n) = f(\tilde{x}_1, \dots, \tilde{x}_n)] \\ &= \frac{p_n}{2^{n-1}} \sum_{k=1}^{2^{n-1}} \sum_{w=0}^{n-1} kp^{n-1-w}(1-p)^w \left(\left| \Omega_{f_0}^{(-2^{n-1}+2k, w)} \right| + \left| \Omega_{f_1}^{(-2^{n-1}+2k, w)} \right| \right). \end{aligned}$$

Proof. Let ℓ be the number of variables (and, without loss of generality, we may assume that these variables are $x_n, x_{n-1}, \dots, x_{n-\ell+1}$), such that $Pr[x_i + \tilde{x}_i] = p_i$, $n - \ell + 1 \leq i \leq n$ and moreover $Pr[x_j + \tilde{x}_j] = p$, $1 \leq j \leq n - \ell$. We will show the result by induction on ℓ . Surely, the case $\ell = 0$ was shown in Theorem 6.

We will show now the $\ell = 1$ case. Thus, $Pr[x_n + \tilde{x}_n = 0] = p_n$ and $Pr[x_i + \tilde{x}_i = 0] = p$. We will be using below the following identity of probabilities of events A, B ,

namely

$$Pr[A] = Pr[A \cap B] + Pr[A \cap \bar{B}] = Pr[A|B]Pr[B] + Pr[A|\bar{B}]Pr[\bar{B}].$$

We write f as a concatenation of two Boolean functions on $n - 1$ variables, say f_0, f_1 , and so, $f(x_1, \dots, x_n) = \bar{x}_n f_0(x_1, \dots, x_{n-1}) + x_n f_1(x_1, \dots, x_{n-1})$. We regard, as usual, f_0, f_1 to be independent. We compute,

$$\begin{aligned} & Pr[f(x_1, \dots, x_n) = f(\tilde{x}_1, \dots, \tilde{x}_n)] \\ &= Pr[f(x_1, \dots, x_n) = f(\tilde{x}_1, \dots, \tilde{x}_n) | x_n = \tilde{x}_n] Pr[x_n = \tilde{x}_n] \\ &\quad + Pr[f(x_1, \dots, x_n) = f(\tilde{x}_1, \dots, \tilde{x}_n) | x_n \neq \tilde{x}_n] Pr[x_n \neq \tilde{x}_n] \\ &= Pr[\bar{x}_n f_0(x_1, \dots, x_{n-1}) + x_n f_1(x_1, \dots, x_{n-1}) \\ &= \bar{x}_n f_0(\tilde{x}_1, \dots, \tilde{x}_{n-1}) + x_n f_1(\tilde{x}_1, \dots, \tilde{x}_{n-1})] p_n \tag{4.13} \\ &\quad + Pr[\bar{x}_n f_0(x_1, \dots, x_{n-1}) + x_n f_1(x_1, \dots, x_{n-1}) \\ &= x_n f_0(\tilde{x}_1, \dots, \tilde{x}_{n-1}) + \bar{x}_n f_1(\tilde{x}_1, \dots, \tilde{x}_{n-1})] (1 - p_n) \\ &= Pr[\bar{x}_n f_0(x_1, \dots, x_{n-1}) + x_n f_1(x_1, \dots, x_{n-1}) \\ &= \bar{x}_n f_0(\tilde{x}_1, \dots, \tilde{x}_{n-1}) + x_n f_1(\tilde{x}_1, \dots, \tilde{x}_{n-1})] p_n, \end{aligned}$$

since the second term is zero, because f_0, f_1 are independent. We now take, as in Theorem 6,

$$\Omega_{f_i}^{(-2^n + 2k, w)} = \{a : \mathcal{C}_{f_i}(a) = -2^n + 2k, wt(a) = w\}, \quad i = 1, 2.$$

Since in the first term of the probability computation (4.13), x_n is a constant now, by using the result of Theorem 6, we obtain

$$\begin{aligned} & Pr[f(x_1, \dots, x_n) = \tilde{f}(x_1, \dots, x_n)] \\ &= (Pr[f_0(x_1, \dots, x_{n-1}) = f_0(\tilde{x}_1, \dots, \tilde{x}_{n-1})] \\ &\quad + Pr[f_1(x_1, \dots, x_{n-1}) = f_1(\tilde{x}_1, \dots, \tilde{x}_{n-1})]) p_n \\ &= \frac{p_n}{2^{n-1}} \sum_{k=1}^{2^{n-1}} \sum_{w=0}^{n-1} k p^{n-1-w} (1-p)^w \left(\left| \Omega_{f_0}^{(-2^{n-1} + 2k, w)} \right| + \left| \Omega_{f_1}^{(-2^{n-1} + 2k, w)} \right| \right). \end{aligned}$$

Now, assuming the claim holds for $(\ell - 1)$, we will show it for ℓ . We split the function

f into 2^ℓ equal length subfunctions,

$$f = f_{0\dots 00} || f_{0\dots 01} || f_{0\dots 10} || \dots || f_{1\dots 11},$$

where the indices are in lexicographical ordering of the k -bit vector space. By the induction assumption, we know that for both f_1 , $i = 0, 1$,

$$\begin{aligned} & Pr[f_i(x_1, \dots, x_{n-1}) = f_i(\tilde{x}_1, \dots, \tilde{x}_{n-1})] \\ &= \frac{\prod_{j=0}^{\ell-1} p_{n-1-j}}{2^{\frac{\ell(2n-\ell-3)}{2}}} \sum_{k=1}^{2^{n-\ell+1}} \sum_{w=0}^{n-1-\ell} \left(\left| \Omega_{f_{i\dots 00}}^{(-2^{n-\ell+1}+2k, w)} \right| + \left| \Omega_{f_{i\dots 01}}^{(-2^{n-\ell+1}+2k, w)} \right| \right. \\ & \quad \left. + \dots + \left| \Omega_{f_{i\dots 11}}^{(-2^{n-\ell+1}+2k, w)} \right| \right). \end{aligned}$$

Using the same computation as above up to the step

$$\begin{aligned} Pr[f(x_1, \dots, x_n) = \tilde{f}(x_1, \dots, x_n)] &= (Pr[f_0(x_1, \dots, x_{n-1}) = f_0(\tilde{x}_1, \dots, \tilde{x}_{n-1})] \\ & \quad + Pr[f_1(x_1, \dots, x_{n-1}) = f_1(\tilde{x}_1, \dots, \tilde{x}_{n-1})]) p_n, \end{aligned}$$

and using the induction assumption on both f_0, f_1 , we obtain our claim. \square

4.4 Conclusion

In this chapter, we have studied the number of Boolean functions class generated from the XOR of two PUFs. Among the two variants of XOR-PUFs, it is shown that the XOR of two n -length Arbiter PUFs with the same input can generate a significant number of Boolean functions. However, the other variant can only generate a negligible number of Boolean functions. We present several findings regarding the existence and non-existence of these functions. Additionally, we provide theoretical proof regarding the probability distribution of output equality from the PUF with a combiner function corresponding to distinct inputs.

Priority Arbiter PUF: Analysis

Contents

5.1	Introduction	92
5.2	Motivation	93
5.3	Characterization of $\mathcal{B}_n^{\text{PA-PUF}}$	93
5.4	Theoretical Estimation of Bias in PA-PUF	99
5.5	Experimental Results and Analysis	113
5.6	Conclusion	120

The research we conducted demonstrates the exploration of Boolean functions obtained by implementing a PA-PUF construction with randomly selected delay parameters. We observe that the class of Boolean functions generated from an n -length PA-PUF is significantly larger than the class of Boolean functions generated from an n -length Arbiter-based PUF. We also study different results related to existence and non-existence of Boolean functions in the set of Boolean functions generated from PA-PUF. Further, we look into the bias estimation of the response bit generated from PA-PUF towards the impact of changing specific bits in the challenge input. Finally, we perform a comparative analysis of PA-PUF to study its cryptographic properties and a detailed analysis of the machine learning attacks.

5.1 Introduction

In 2020, Roy et al. [RRM21] looked into the prime reason behind the biases observed by Siddhanti et al. [SBC+19b]. Roy et al. have observed that an n -length Arbiter based PUF with two paths cannot generate all possible 2^{2^n} Boolean functions. In fact they have shown that the number of Boolean functions generated from an n -length PUF are 4, 14, 104 and 1882 for $n = 1, 2, 3, 4$ respectively. Due to this limitation, we observe biases in the output bits from a classical Arbiter based PUF. They have also shown different characteristics of Boolean functions generated from an n -length Arbiter based PUF with two paths.

From the analytical results and multiple weaknesses of Arbiter-based PUF with two paths, the question which comes to our mind is that what will happen if we increase the number of paths in PUF? In other words, is it possible to design an Arbiter-based PUF with more than two paths?

In 2022, Singh et al. [SBP+22] have proposed a modified construction of arbiter-based PUF with three paths (Top path, Center path and Bottom path), namely Priority Arbiter PUF (PA-PUF). Figure 5-1 shows the hardware implementation of the three paths using multiplexers. They have provided an experimental analysis on their proposed design with enhanced robustness though, a mathematical analysis is not carried out. We study PA-PUF in detail in this chapter.

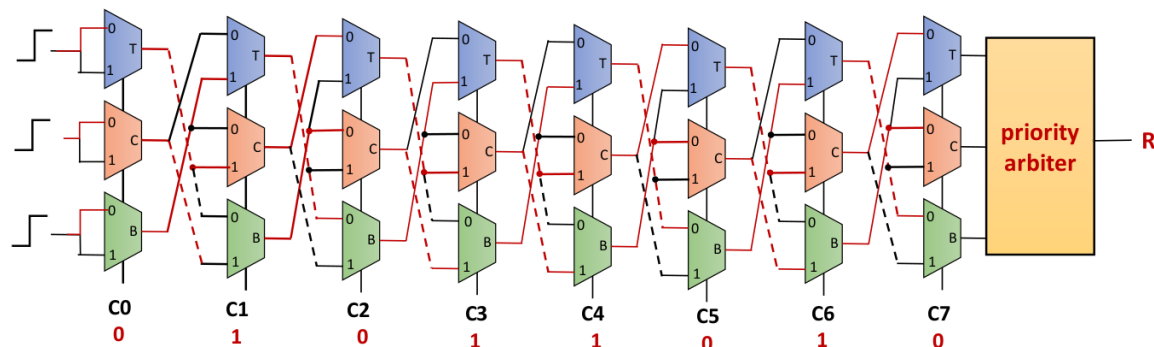


Figure 5-1: Schematic of proposed PA-PUF which includes three parallel multiplexer lines (T, C, and B) and a priority arbiter at the end to generate the response bit. Challenges are considered as 0/1 instead of 1/-1.

5.2 Motivation

Arbiter based PUF models are analyzed in multiple articles. It has been observed that the output from single Arbiter based PUF is not pseudorandom. In fact, Siddhanti et al. [SBC⁺19b] have shown that there is a significant amount of non-randomness in the two output bits from an Arbiter based PUF corresponding to two inputs which differ at particular positions. In [RRM21] Roy et al. have shown that $\mathcal{B}_n^{\text{PUF}}$ is significantly smaller than \mathcal{B}_n . In [RRM21] it has also been shown that several classes of Boolean functions do exist in $\mathcal{B}_n^{\text{PUF}}$. Due to these issues non-randomness exists in the output bit from an Arbiter based PUF. Now to improve the cryptographic properties multiple authors have proposed constructions of Arbiter based PUF. In most of these constructions designers have used multiple Arbiter based PUFs to either generate a combined output bit or they have fed output from one PUF to another to generate the final output bit. In recent time Singh et al. [SBP⁺22] have proposed Priority Arbiter PUF. Here the authors have used three paths instead of two paths. The design of PA-PUF looks promising towards solving cryptographic (specifically non-randomness) issues present in the classical Arbiter based PUF. There is no such article which tries to analyze the design of PA-PUF. In this chapter, we consider PA-PUF and study its cryptographic properties.

5.3 Characterization of $\mathcal{B}_n^{\text{PA-PUF}}$

We already know that an n -length PA-PUF takes input $C = \{c[0], c[1], \dots, c[n-1]\}$ from $\{-1, 1\}^n$ and produces 0/1 as an output. To compute $|\mathcal{B}_n^{\text{PA-PUF}}|$, we start with $n = 1, 2$. For finding the $\mathcal{B}_n^{\text{PA-PUF}}$ we randomly choose the delay parameters of the top, center and bottom paths from normal distributions and generate the outputs from the PA-PUF for all possible challenge inputs. We vary the delay parameters and generate the outputs for all possible challenge inputs. Finally count the distinct such truth tables to determine $\mathcal{B}_n^{\text{PA-PUF}}$.

Proposition 10. *All possible Boolean functions involving 1-variable and 2-variable can be generated using 1-length and 2-length PA-PUF respectively i.e., $\mathcal{B}_1^{\text{PA-PUF}} = \mathcal{B}_1$ and $\mathcal{B}_2^{\text{PA-PUF}} = \mathcal{B}_2$.*

Proof. This proposition can be shown true by exhaustively enumerating $\mathcal{B}_1^{\text{PA-PUF}}$ and

$\mathcal{B}_2^{\text{PA-PUF}}$. We have considered 1-length and 2-length PA-PUFs for different random delay parameters and observed that all the possible truth tables are generated in our experiment. Hence $\mathcal{B}_1^{\text{PA-PUF}} = \mathcal{B}_1$, $\mathcal{B}_2^{\text{PA-PUF}} = \mathcal{B}_2$. \square

In [RRM21] it has been shown that $|\mathcal{B}_n^{\text{PUF}}| = 4, 14$ for $n = 1, 2$ respectively (see Proposition 1, 2 of [RRM21]). Two Boolean functions $f_1 = (0\ 1\ 0\ 1)$ and $f_2 = 1 \oplus f_1$ do not belong to $\mathcal{B}_2^{\text{PUF}}$. Hence $|\mathcal{B}_2^{\text{PA-PUF}}| > |\mathcal{B}_2^{\text{PUF}}|$. With this connection in Proposition 11 we prove why $f_1, f_2 \in \mathcal{B}_2^{\text{PA-PUF}}$. For proving this we find the expressions of $\Delta_{BT}(0)$, $\Delta_{BT}(1)$. From Equation (2.6) we get,

$$\begin{aligned}\Delta_{BT}(0) &= \frac{c[0]}{2} \cdot \left(p_0^{(B)} - q_0^{(T)} - q_0^{(B)} + r_0^{(T)} \right) + \frac{1}{2} \cdot \left(p_0^{(B)} - q_0^{(T)} + q_0^{(B)} - r_0^{(T)} \right), \\ \Delta_{CB}(0) &= \frac{c[0]}{2} \cdot \left(r_0^{(C)} - p_0^{(B)} - p_0^{(C)} + q_0^{(B)} \right) + \frac{1}{2} \cdot \left(r_0^{(C)} - p_0^{(B)} + p_0^{(C)} - q_0^{(B)} \right), \\ \Delta_{TC}(0) &= \frac{c[0]}{2} \cdot \left(q_0^{(T)} - r_0^{(C)} - r_0^{(T)} + p_0^{(C)} \right) + \frac{1}{2} \cdot \left(q_0^{(T)} - r_0^{(C)} + r_0^{(T)} - p_0^{(C)} \right).\end{aligned}$$

Let $\alpha = \frac{p+q}{2}, \beta = \frac{p-q}{2}, \gamma = \frac{q+r}{2}, \delta = \frac{q-r}{2}, \kappa = \frac{r+p}{2}$ and $\lambda = \frac{r-p}{2}$. Then,

$$\begin{aligned}\Delta_{BT}(0) &= c[0] \cdot \left(\beta_0^{(B)} - \delta_0^{(T)} \right) + \left(\alpha_0^{(B)} - \gamma_0^{(T)} \right), \\ \Delta_{CB}(0) &= c[0] \cdot \left(\lambda_0^{(C)} - \beta_0^{(B)} \right) + \left(\kappa_0^{(C)} - \alpha_0^{(B)} \right), \\ \Delta_{TC}(0) &= c[0] \cdot \left(\delta_0^{(T)} - \lambda_0^{(C)} \right) + \left(\gamma_0^{(T)} - \kappa_0^{(C)} \right)\end{aligned}$$

and

$$\begin{aligned}\Delta_{BT}(1) &= \frac{c[0]}{2} \cdot \left(\delta_0^{(T)} - \beta_0^{(B)} \right) + \frac{1}{2} \cdot \left(\gamma_0^{(T)} - \alpha_0^{(B)} \right) \\ &\quad + \frac{c[0] \cdot c[1]}{2} \cdot \left(\delta_0^{(T)} - 2\lambda_0^{(C)} + \beta_0^{(B)} \right) \\ &\quad + \frac{c[1]}{2} \cdot \left(\gamma_0^{(T)} - 2\kappa_0^{(C)} + \alpha_0^{(B)} \right) + c[1] \cdot \left(\beta_1^{(B)} - \delta_1^{(T)} \right) + \left(\alpha_1^{(B)} - \gamma_1^{(T)} \right).\end{aligned}\tag{5.1}$$

We use the expression of $\Delta_{BT}(1)$ as in Equation (5.1) to prove that $f_1 = (0\ 1\ 0\ 1)$ and $f_2 = 1 \oplus f_1$ belong to $\mathcal{B}_2^{\text{PUF}}$ in Proposition 11.

Proposition 11. *Two Boolean functions $f_1 = (0\ 1\ 0\ 1)$ and $f_2 = (1\ 0\ 1\ 0)$ do not belong to $\mathcal{B}_2^{\text{PUF}}$, but $f_1, f_2 \in \mathcal{B}_2^{\text{PA-PUF}}$.*

Proof. We will be using Equation (5.1) for 2-length PA-PUF. We consider the truth table of f_1 first.

C_1	C_0	$\Delta_{BT}(1)$
1	1	$\delta_0^{(T)} + \gamma_0^{(T)} - \lambda_0^{(C)} - \kappa_0^{(C)} + \beta_1^{(B)} - \delta_1^{(T)} + \alpha_1^{(B)} - \gamma_1^{(T)} > 0$
1	-1	$-\delta_0^{(T)} + \gamma_0^{(T)} + \lambda_0^{(C)} - \kappa_0^{(C)} + \beta_1^{(B)} - \delta_1^{(T)} + \alpha_1^{(B)} - \gamma_1^{(T)} < 0$
-1	1	$-\beta_0^{(B)} - \alpha_0^{(B)} + \lambda_0^{(C)} + \kappa_0^{(C)} - \beta_1^{(B)} + \delta_1^{(T)} + \alpha_1^{(B)} - \gamma_1^{(T)} > 0$
-1	-1	$\beta_0^{(B)} - \alpha_0^{(B)} - \lambda_0^{(C)} + \kappa_0^{(C)} - \beta_1^{(B)} + \delta_1^{(T)} + \alpha_1^{(B)} - \gamma_1^{(T)} < 0$

If there exists a solution for which the above condition holds, then we can say that the truth values of f_1 can be generated. If we add two > 0 inequalities, then we have $\left(\delta_0^{(T)} - \beta_0^{(B)} - \alpha_0^{(B)} + \gamma_0^{(T)} + 2 \cdot \left(\alpha_1^{(B)} - \gamma_1^{(T)}\right)\right) > 0$ and if we add two < 0 inequalities, we have $\left(-\delta_0^{(T)} + \beta_0^{(B)} - \alpha_0^{(B)} + \gamma_0^{(T)} + 2 \cdot \left(\alpha_1^{(B)} - \gamma_1^{(T)}\right)\right) < 0$. Let $A = \left(\delta_0^{(T)} - \beta_0^{(B)}\right)$ and $B = \left(-\alpha_0^{(B)} + \gamma_0^{(T)} + 2 \cdot \left(\alpha_1^{(B)} - \gamma_1^{(T)}\right)\right)$. Hence we have $A+B > 0$ and $-A+B < 0$. This condition holds for some A and B i.e., not producing any contradictory statement. Hence the truth table of f_1 can be generated using 2-length PA-PUF i.e., $f_1 \in \mathcal{B}_2^{\text{PA-PUF}}$. Similarly, we can also show that $f_2 \in \mathcal{B}_2^{\text{PA-PUF}}$. \square

We are now interested to investigate the Lemma 1 from Chapter 3 for $\mathcal{B}_n^{\text{PA-PUF}}$.

Proposition 12. *For any n -variable Boolean function if $f \in \mathcal{B}_n^{\text{PA-PUF}}$ then $(1 \oplus f) \in \mathcal{B}_n^{\text{PA-PUF}}$.*

Proof. Let $f \in \mathcal{B}_n^{\text{PA-PUF}}$. We know that depending on the sign of $\Delta_{BT}(n)$ with delay parameters $\alpha_i^{(B)}, \beta_i^{(B)}, \gamma_i^{(T)}, \delta_i^{(T)}, \kappa_i^{(C)}$ and $\lambda_i^{(C)}$ (for $i = 0, 1, \dots, n$) the truth table of $(1 \oplus f)$ is generated. Using Proposition 1, an $(n+1)$ -length PA-PUF can be written as $\Delta_{BT}(n) = -\frac{1}{2} \cdot \Delta_{BT}(n-1) + \frac{c[n]}{2} \cdot (\Delta_{TC}(n-1) - \Delta_{CB}(n-1)) + c[n] \cdot \left(\beta_n^{(B)} - \delta_n^{(T)}\right) + \left(\alpha_n^{(B)} - \gamma_n^{(T)}\right)$. Now if we construct an $(n+1)$ -length PA-PUF $\Delta'_{BT}(n)$ with delay parameters $\alpha_i'^{(B)} = -\alpha_i^{(B)}, \beta_i'^{(B)} = -\beta_i^{(B)}, \gamma_i'^{(T)} = -\gamma_i^{(T)}, \delta_i'^{(T)} = -\delta_i^{(T)}, \kappa_i'^{(C)} = -\kappa_i^{(C)}$ and $\lambda_i'^{(C)} = -\lambda_i^{(C)}$ for $i = 0, 1, 2, \dots, n$; then the sign of $\Delta'_{BT}(n)$ and the sign of $\Delta_{BT}(n)$ will be opposite for the same challenge values. Hence the truth table of $(1 \oplus f)$ can be generated from $\Delta'_{BT}(n)$. Hence $(1 \oplus f) \in \mathcal{B}_n^{\text{PA-PUF}}$ whenever $f \in \mathcal{B}_n^{\text{PA-PUF}}$. \square

Now we will move towards the case for $n = 3$. We first write down the expression of $\Delta_{BT}(2)$ in terms of $c[0], c[1], c[2]$. Using the expression of $\Delta_{BT}(2)$ we prove the

non-existence of one Boolean function in the set $\mathcal{B}_3^{\text{PA-PUF}}$ in Proposition 13.

$$\begin{aligned}
\Delta_{BT}(2) &= \frac{c[0]}{2^2} \cdot (\beta_0^{(B)} - \delta_0^{(T)}) + \frac{1}{2^2} \cdot (\alpha_0^{(B)} - \gamma_0^{(T)}) \\
&+ \frac{c[0] \cdot c[1]}{2^2} \cdot (2\lambda_0^{(C)} - \delta_0^{(T)} - \beta_0^{(B)}) + \frac{c[1]}{2^2} \cdot (2\kappa_0^{(C)} - \gamma_0^{(T)} - \alpha_0^{(B)}) \\
&+ \frac{c[1]}{2} \cdot (\delta_1^{(T)} - \beta_1^{(B)}) + \frac{1}{2} \cdot (\gamma_1^{(T)} - \alpha_1^{(B)}) \\
&+ \frac{c[0] \cdot c[2]}{2^2} \cdot (2\lambda_0^{(C)} - \delta_0^{(T)} - \beta_0^{(B)}) + \frac{c[2]}{2^2} \cdot (2\kappa_0^{(C)} - \gamma_0^{(T)} - \alpha_0^{(B)}) \quad (5.2) \\
&+ \frac{c[0] \cdot c[1] \cdot c[2]}{2^2} \cdot 3 (\delta_0^{(T)} - \beta_0^{(B)}) + \frac{c[1] \cdot c[2]}{2^2} \cdot 3 (\gamma_0^{(T)} - \alpha_0^{(B)}) \\
&+ \frac{c[1] \cdot c[2]}{2} \cdot (\delta_1^{(T)} - 2\lambda_1^{(C)} + \beta_1^{(B)}) + \frac{c[2]}{2} \cdot (\gamma_1^{(T)} - 2\kappa_1^{(C)} + \alpha_1^{(B)}) \\
&+ c[2] \cdot (\beta_2^{(B)} - \delta_2^{(T)}) + (\alpha_2^{(B)} - \gamma_2^{(T)}).
\end{aligned}$$

Proposition 13. *The function $f_1 = (0\ 0\ 0\ 1\ 1\ 0\ 0\ 0) \notin \mathcal{B}_3^{\text{PA-PUF}}$.*

Proof. We consider the truth table of $f_1 = (0\ 0\ 0\ 1\ 1\ 0\ 0\ 0)$ and the Equation (5.2) for a 3-length PA-PUF. Let $X = (\beta_2^{(B)} - \delta_2^{(T)}) + (\alpha_2^{(B)} - \gamma_2^{(T)})$ and $Y = -(\beta_2^{(B)} - \delta_2^{(T)}) + (\alpha_2^{(B)} - \gamma_2^{(T)})$.

$c[2]$	$c[1]$	$c[0]$	$\Delta_{BT}(2)$
1	1	1	$-4\alpha_0^{(B)} - 4\beta_0^{(B)} + 4\kappa_0^{(C)} + 4\lambda_0^{(C)} + 4\delta_1^{(T)} + 4\gamma_1^{(T)} - 4\kappa_1^{(C)} - 4\lambda_1^{(C)} + 4X > 0$
1	1	-1	$-4\alpha_0^{(B)} + 4\beta_0^{(B)} + 4\kappa_0^{(C)} - 4\lambda_0^{(C)} + 4\delta_1^{(T)} + 4\gamma_1^{(T)} - 4\kappa_1^{(C)} - 4\lambda_1^{(C)} + 4X > 0$
1	-1	1	$4\alpha_0^{(B)} + 4\beta_0^{(B)} - 4\gamma_0^{(T)} - 4\delta_0^{(T)} - 4\delta_1^{(T)} + 4\gamma_1^{(T)} - 4\kappa_1^{(C)} + 4\lambda_1^{(C)} + 4X > 0$
1	-1	-1	$4\alpha_0^{(B)} - 4\beta_0^{(B)} - 4\gamma_0^{(T)} + 4\delta_0^{(T)} - 4\delta_1^{(T)} + 4\gamma_1^{(T)} - 4\kappa_1^{(C)} + 4\lambda_1^{(C)} + 4X < 0$
-1	1	1	$4\alpha_0^{(B)} + 4\beta_0^{(B)} - 4\gamma_0^{(T)} - 4\delta_0^{(T)} - 4\alpha_1^{(B)} - 4\beta_1^{(B)} + 4\kappa_1^{(C)} + 4\lambda_1^{(C)} + 4Y < 0$
-1	1	-1	$4\alpha_0^{(B)} - 4\beta_0^{(B)} - 4\gamma_0^{(T)} + 4\delta_0^{(T)} - 4\alpha_1^{(B)} - 4\beta_1^{(B)} + 4\kappa_1^{(C)} + 4\lambda_1^{(C)} + 4Y > 0$
-1	-1	1	$-4\lambda_0^{(C)} - 4\kappa_0^{(C)} + 4\gamma_0^{(T)} + 4\delta_0^{(T)} - 4\alpha_1^{(B)} + 4\beta_1^{(B)} + 4\kappa_1^{(C)} - 4\lambda_1^{(C)} + 4Y > 0$
-1	-1	-1	$4\lambda_0^{(C)} - 4\kappa_0^{(C)} + 4\gamma_0^{(T)} - 4\delta_0^{(T)} - 4\alpha_1^{(B)} + 4\beta_1^{(B)} + 4\kappa_1^{(C)} - 4\lambda_1^{(C)} + 4Y > 0$

From the above system of inequalities we get $\Delta_{BT}(2)\Big|_{(1\ -1\ 1)} - \Delta_{BT}(2)\Big|_{(1\ -1\ -1)} = 8 \cdot (\beta_0^{(B)} - \delta_0^{(T)}) > 0$ and $\Delta_{BT}(2)\Big|_{(-1\ 1\ -1)} - \Delta_{BT}(2)\Big|_{(-1\ 1\ 1)} = 8 \cdot (\delta_0^{(T)} - \beta_0^{(B)}) > 0$. This generates a contradiction. Hence the truth table of $f_1 = (0\ 0\ 0\ 1\ 1\ 0\ 1\ 0)$ can not be generated from a 3-length PA-PUF. \square

Proposition 13 shows that $\mathcal{B}_3^{\text{PA-PUF}} \subset \mathcal{B}_3$. We now choose randomly many delay parameters and construct PA-PUF for $n = 3, 4$. To each of these PA-PUFs we give all possible challenge $C \in \{-1, 1\}^n$ as input and generate the truth tables. Finally count the distinct number of truth tables to find $|\mathcal{B}_n^{\text{PA-PUF}}|$. The experimental

n	$ \mathcal{B}_n^{\text{PUF}} $	$ \mathcal{B}_n^{\text{PA-PUF}} $
1	4	4
2	14	16
3	104	214
4	1882	16584

Table 5.1: Comparison between $\mathcal{B}_n^{\text{PA-PUF}}$ and $\mathcal{B}_n^{\text{PUF}}$.

observation for different values of n is given in Table 5.1. We know that any $(n + 1)$ -variable Boolean function f can be expressed as $f(x_0, x_1, \dots, x_n) = (1 \oplus x_n) \cdot f_1(x_0, x_1, \dots, x_{n-1}) \oplus x_n \cdot f_2(x_0, x_1, \dots, x_{n-1})$, where f_1, f_2 are two n -variable Boolean functions. The truth table of f will be $f_1 \parallel f_2$ i.e., concatenation of the truth tables of f_1 and f_2 . From our analysis on $\mathcal{B}_2^{\text{PA-PUF}}$ and $\mathcal{B}_3^{\text{PA-PUF}}$ we have observed that $|\mathcal{B}_2^{\text{PA-PUF}}| = 16$ and $|\mathcal{B}_3^{\text{PA-PUF}}| = 214$. If we concatenate every $f_1, f_2 \in \mathcal{B}_2^{\text{PA-PUF}}$ and generate Boolean functions then we should have 256 number of Boolean functions but $|\mathcal{B}_3^{\text{PA-PUF}}| = 214$. That means every $f_1 \parallel f_2$ may not belong to $\mathcal{B}_3^{\text{PA-PUF}}$. In fact if we take $f_1 = (0\ 0\ 0\ 1), f_2 = (1\ 0\ 0\ 0) \in \mathcal{B}_2^{\text{PA-PUF}}$ and construct $f = f_1 \parallel f_2$ we get the function $f = (0\ 0\ 0\ 1\ 1\ 0\ 0\ 0)$ and our Proposition 13 says that $f \notin \mathcal{B}_3^{\text{PA-PUF}}$. Thus in general we can say that even if $f_1, f_2 \in \mathcal{B}_n^{\text{PA-PUF}}$ it is not guaranteed that $f_1 \parallel f_2$ will lie in the set $\mathcal{B}_{n+1}^{\text{PA-PUF}}$ for every $n \geq 2$. Thus the characterization of $\mathcal{B}_n^{\text{PA-PUF}}$ does not follow the same path as \mathcal{B}_n . Following theorem which characterizes $\mathcal{B}_n^{\text{PA-PUF}}$ for $n = 2, 3, 4$ follows directly from Table 5.1.

Theorem 8. $|\mathcal{B}_n| = |\mathcal{B}_n^{\text{PA-PUF}}| > |\mathcal{B}_n^{\text{PUF}}|$ for $n = 2$ and $|\mathcal{B}_n| > |\mathcal{B}_n^{\text{PA-PUF}}| > |\mathcal{B}_n^{\text{PUF}}|$ for $n = 3, 4$.

Next in Theorem 9 we find the generalization of the Theorem 8.

Theorem 9. $\mathcal{B}_n^{\text{PUF}} \subseteq \mathcal{B}_n^{\text{PA-PUF}}$ for every $n > 0$.

Proof. The experimental results given in Table 5.1 validate the theorem for $n = 1, 2, 3, 4$. For general value of n we look into the mathematical expressions of delay difference between the two paths in PUF and delay differences between the top and bottom paths of PA-PUF. The expression of the delay difference between two paths for PUF is $\Delta(n-1) = c[n-1]\Delta(n-2) + \alpha_{n-1}c[n-1] + \beta_{n-1}$. Here $\alpha_{n-1} = \frac{p_{n-1}-q_{n-1}}{2} + \frac{r_{n-1}-s_{n-1}}{2}$ and $\beta_{n-1} = \frac{p_{n-1}-q_{n-1}}{2} - \frac{r_{n-1}-s_{n-1}}{2}$ (see [SBC⁺19b]). For $c[n-1] = 1$ the delay difference becomes $\Delta(n-1) = \Delta(n-2) + p_{n-1} - q_{n-1}$. For $c[n-1] = -1$ the delay difference becomes $\Delta(n-1) = -\Delta(n-2) - (r_{n-1} - s_{n-1})$. Depending upon the sign of

$\Delta(n-1)$ the PUF outputs either 0 or 1. Whereas the PA-PUF outputs either 0 or 1 depending upon the sign of $\Delta_{BT}(n-1)$. For $c[n-1] = 1$, the expression of $\Delta_{BT}(n-1)$ becomes $\Delta_{BT}(n-1) = \Delta_{TC}(n-2) + (p_{n-1}^{(B)} - q_{n-1}^{(T)})$ and for $c[n-1] = -1$ the expression of $\Delta_{BT}(n-1)$ becomes $\Delta_{BT}(n-1) = \Delta_{CB}(n-2) + (q_{n-1}^{(B)} - r_{n-1}^{(T)})$. It can be noticed that under the special condition $\Delta(n-2) = \Delta_{TC}(n-2)$, $p_{n-1} = p_{n-1}^{(B)}$, $q_{n-1} = q_{n-1}^{(T)}$ for $c[n-1] = 1$ and $\Delta(n-2) = -\Delta_{CB}(n-2)$, $r_{n-1} = q_{n-1}^{(B)}$, $s_{n-1} = r_{n-1}^{(T)}$ for $c[n-1] = -1$ the last stage of PUF can be replicated in the last stage of PA-PUF. Thus the output from the PUF will be exactly same as the output from the PA-PUF. It can also be noted that this is a special condition, except this condition there are many conditions under which an n -length PUF will produce same output as an n -length PA-PUF. Hence we can say that if $f \in \mathcal{B}_n^{\text{PUF}}$ then $f \in \mathcal{B}_n^{\text{PA-PUF}}$. Thus $\mathcal{B}_n^{\text{PUF}} \subseteq \mathcal{B}_n^{\text{PA-PUF}}$. The opposite relation will not hold i.e., $\mathcal{B}_n^{\text{PA-PUF}} \not\subseteq \mathcal{B}_n^{\text{PUF}}$ because of the involvement of random $\Delta_{TC}(n-2)$ and $\Delta_{CB}(n-2)$ in the expression of $\Delta_{BT}(n-2)$. \square

Now we are in a position to determine the set $\mathcal{B}_n^{\text{PUF}}$ for any $n \in \mathbb{N}$. Till now to generate the set $\mathcal{B}_n^{\text{PA-PUF}}$ we have followed an exhaustive search technique i.e., we have randomly taken the delay parameters to construct a PA-PUF. We will now propose an algorithm which will deterministically determine whether $f \in \mathcal{B}_n^{\text{PA-PUF}}$ or not for every Boolean function $f \in \mathcal{B}_n$.

We have implemented Algorithm 4 in SageMath [sag] for different values of n . The experimental observation is given in Table 5.2. The experiment is performed in a laptop with with a processor of 2.80GHz clock, 16 GB RAM and Linux (Ubuntu 23.04) environment.

n	f (in hex)	Decision	Time Required (in sec)
2	$f = \mathbf{t0x5}$	$f \in \mathcal{B}_2^{\text{PA-PUF}}$	1.16
3	$f = \mathbf{t0x19}$	$f \notin \mathcal{B}_3^{\text{PA-PUF}}$	2.002
4	$f = \mathbf{t0x3FD}$	$f \in \mathcal{B}_4^{\text{PA-PUF}}$	10.09
5	$f = \mathbf{t0xE7}$	$f \notin \mathcal{B}_5^{\text{PA-PUF}}$	75.65

Table 5.2: Experimental validation of Algorithm 4.

Algorithm 4: Deterministic determination of $\mathcal{B}_n^{\text{PA-PUF}}$

```
Input :  $f \in \mathcal{B}_n$ 
Output:  $f \in \mathcal{B}_n^{\text{PA-PUF}}$  or  $f \notin \mathcal{B}_n^{\text{PA-PUF}}$ 
1 Sys = {};
2 Consider the mathematical formulae of  $\Delta_{CB}(n-1)$ ,  $\Delta_{TC}(n-1)$  and
    $\Delta_{BT}(n-1)$  as per the Equations (2.4), (2.5), (2.6);
3 for each  $C \in \{-1, 1\}^n$  do
4   | if  $f(C) = 1$  then
5   |   | Eqn  $\leftarrow \Delta_{BT}(n-1) > 0$ ;
6   |   | // Delay parameters are the unknown
7   | end
8   | else
9   |   | Eqn  $\leftarrow \Delta_{BT}(n-1) < 0$ ;
10  |   | // Delay parameters are the unknown
11  | end
12  | Sys = Sys  $\cup$  {Eqn};
13 end
14 if  $S \neq \{\}$  then
15 |   |  $R \leftarrow f \in \mathcal{B}_n^{\text{PA-PUF}}$ ;
16 end
17 else
18 |   |  $R \leftarrow f \notin \mathcal{B}_n^{\text{PA-PUF}}$ ;
19 end
20 return  $R$ ;
```

5.4 Theoretical Estimation of Bias in PA-PUF

In this section we look into the bias in the output bit generated from an n -length PA-PUF when certain input bit is flipped i.e., if z_C is the output bit corresponding to the challenge C and $z_{\tilde{C}}$ is the challenge \tilde{C} , we look forward to find the probability $Pr[z_C = z_{\tilde{C}}]$ where C and \tilde{C} differ at only one position. We consider two cases: (1) C and \tilde{C} differ at $(n-1)$ -th position i.e., last position, (2) C and \tilde{C} differ at

$(n - 2)$ -th position i.e., second last position. The estimation of $Pr[z_C = z_{\bar{C}}]$ for normal Arbiter based PUF was reported by Siddhanti et al. [SBC⁺19b]. First we perform experiments to compare the probabilities $Pr[z_C = z_{\bar{C}}]$ between PUF and PA-PUF for both the above mentioned types of challenge inputs. Our experiment shows significant improvements in the probability $Pr[z_C = z_{\bar{C}}]$ for PA-PUF over the classical Arbiter based PUF (see Figure 5-2).

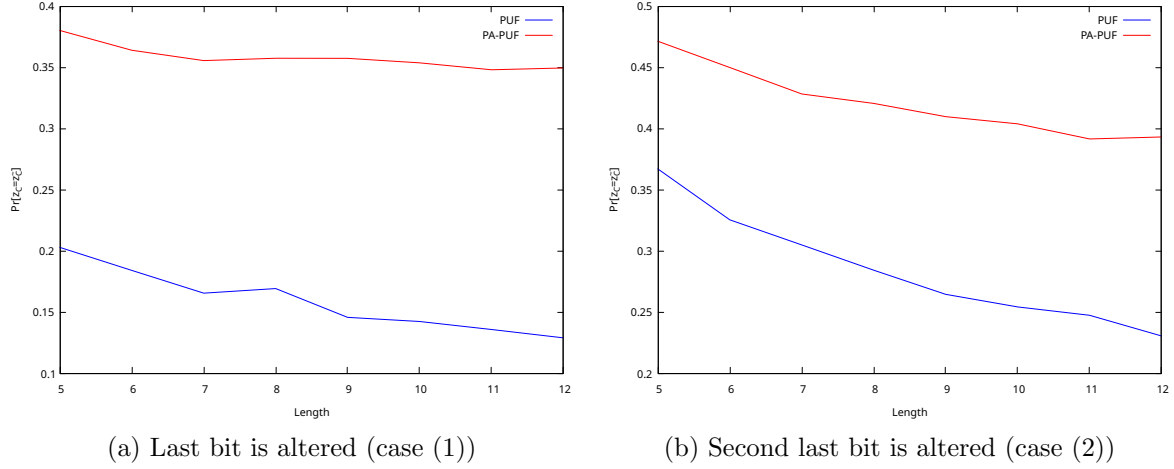


Figure 5-2: Comparison of the restricted autocorrelation.

This experimental observation motivates us to find the mathematical expressions of the probabilities $Pr[z_C = z_{\bar{C}}]$ for the above two cases. Before going to the estimation of the probabilities, we first need the distributions of $\Delta_{BT}(n)$, $\Delta_{TC}(n)$ and $\Delta_{CB}(n)$ for $n \geq 0$.

Theorem 10. $\Delta_{BT}(n), \Delta_{CB}(n) \sim \mathcal{N}(0, \sqrt{n+1}\sigma)$ and $\Delta_{TC}(n) \sim \mathcal{N}(0, \sqrt{2(n+1)}\sigma)$.

Proof. The mathematical expressions of $\Delta_{BT}(0)$, $\Delta_{TC}(0)$ and $\Delta_{CB}(0)$ are as follows:

$$\begin{aligned}
\Delta_{BT}(0) &= \frac{c[0]}{2} (p_0^{(B)} - q_0^{(T)} - q_0^{(B)} + r_0^{(T)}) + \frac{1}{2} (p_0^{(B)} - q_0^{(T)} + q_0^{(B)} - r_0^{(T)}) \\
&= \frac{1+c[0]}{2} (p_0^{(B)} - q_0^{(T)}) + \frac{1-c[0]}{2} (q_0^{(B)} - r_0^{(T)}), \\
\Delta_{CB}(0) &= \frac{c[0]}{2} (r_0^{(C)} - p_0^{(B)} - p_0^{(C)} + q_0^{(B)}) + \frac{1}{2} (r_0^{(C)} - p_0^{(B)} + p_0^{(C)} - q_0^{(B)}) \\
&= \frac{1+c[0]}{2} (r_0^{(C)} - p_0^{(B)}) + \frac{1-c[0]}{2} (p_0^{(C)} - q_0^{(B)}), \\
\Delta_{TC}(0) &= \frac{c[0]}{2} (q_0^{(T)} - r_0^{(C)} - r_0^{(T)} + p_0^{(C)}) + \frac{1}{2} (q_0^{(T)} - r_0^{(C)} + r_0^{(T)} - p_0^{(C)}) \\
&= \frac{1+c[0]}{2} (q_0^{(T)} - r_0^{(C)}) + \frac{1-c[0]}{2} (r_0^{(T)} - p_0^{(C)}).
\end{aligned}$$

As $p_i^{(j)}, q_i^{(j)}, r_i^{(j)} \sim \mathcal{N}(\mu, \sigma)$, $\Delta_{BT}(0) \sim \mathcal{N}(0, \sigma_1)$ where

$$\sigma_1 = \sqrt{\left(\frac{1+c[0]}{2}\right)^2 \sigma^2 + \left(\frac{1-c[0]}{2}\right)^2 \sigma^2} = \sigma \sqrt{\frac{2(c[0]^2 + 1)}{4}} = \sigma$$

Hence $\Delta_{BT}(0) \sim \mathcal{N}(0, \sigma)$. Similarly it can be shown that $\Delta_{CB}(0) \sim \mathcal{N}(0, \sigma)$. Now from Proposition 1 we know that $\Delta_{TC}(0) = -(\Delta_{BT}(0) + \Delta_{CB}(0))$. Thus $\Delta_{TC}(0)$ will follow $\mathcal{N}(0, \sqrt{2}\sigma)$. Now we will consider the expressions of $\Delta_{BT}(1), \Delta_{CB}(1)$, and $\Delta_{TC}(1)$ to determine their distributions.

$$\begin{aligned} \Delta_{BT}(1) &= \frac{1}{2}(\Delta_{TC}(0) + \Delta_{CB}(0)) + \frac{c[1]}{2}(\Delta_{TC}(0) - \Delta_{CB}(0)) \\ &\quad + \frac{c[1]}{2}(p_1^{(B)} - q_1^{(T)} - q_1^{(B)} + r_1^{(T)}) + \frac{1}{2}(p_1^{(B)} - q_1^{(T)} + q_1^{(B)} - r_1^{(T)}) \\ &= -\frac{1}{2}\Delta_{BT}(0) + \frac{c[1]}{2}(\Delta_{TC}(0) - \Delta_{CB}(0)) + \frac{1+c[1]}{2}(p_1^{(B)} - q_1^{(T)}) \\ &\quad + \frac{1-c[1]}{2}(q_1^{(B)} - r_1^{(T)}), \\ \Delta_{CB}(1) &= \frac{1}{2}(\Delta_{BT}(0) + \Delta_{TC}(0)) + \frac{c[1]}{2}(\Delta_{BT}(0) - \Delta_{TC}(0)) \\ &\quad + \frac{c[1]}{2}(r_1^{(C)} - p_1^{(B)} - p_1^{(C)} + q_1^{(B)}) + \frac{1}{2}(r_1^{(C)} - p_1^{(B)} + p_1^{(C)} - q_1^{(B)}) \\ &= -\frac{1}{2}\Delta_{CB}(0) + \frac{c[1]}{2}(\Delta_{BT}(0) - \Delta_{TC}(0)) + \frac{1+c[1]}{2}(r_1^{(C)} - p_1^{(B)}) \\ &\quad + \frac{1-c[1]}{2}(p_1^{(C)} - q_1^{(B)}), \\ \Delta_{TC}(1) &= -(\Delta_{BT}(1) + \Delta_{CB}(1)). \end{aligned}$$

Here $\Delta_{BT}(0), \Delta_{CB}(0)$ follow $\mathcal{N}(0, \sigma)$ and $\Delta_{TC}(0)$ follows $\mathcal{N}(0, \sqrt{2}\sigma)$. Hence the mean of the distribution of $\Delta_{BT}(1)$ will be 0 and it will follow $\mathcal{N}(0, \sigma_2)$ where

$$\begin{aligned} \sigma_2 &= \sqrt{\frac{1}{4}\sigma^2 + \frac{1}{4}(\sigma^2 + 2\sigma^2) + \frac{(1+c[1])^2}{4}(\sigma^2 + \sigma^2) + \frac{(1-c[1])^2}{4}(\sigma^2 + \sigma^2)} \\ &= 2\sigma^2. \end{aligned}$$

Hence $\Delta_{BT}(1)$ follows $\mathcal{N}(0, \sqrt{2}\sigma)$. Similarly it can be shown that $\Delta_{CB}(1)$ follows $\mathcal{N}(0, \sqrt{2}\sigma)$. Finally $\Delta_{TC}(1)$ will follow $\mathcal{N}(0, 2\sigma)$ as $\Delta_{TC}(1) = -(\Delta_{BT}(1) + \Delta_{CB}(1))$. By doing similar computation we get $\Delta_{BT}(2), \Delta_{CB}(2) \sim \mathcal{N}(0, \sqrt{3}\sigma)$ and $\Delta_{TC}(2) \sim \mathcal{N}(0, \sqrt{6}\sigma)$.

Now we assume that $\Delta_{BT}(i), \Delta_{CB}(i)$ follow $\mathcal{N}(0, \sqrt{i+1}\sigma)$ and $\Delta_{TC}(i)$ follow $\mathcal{N}(0, \sqrt{2(i+1)}\sigma)$. With this hypothesis we derive the distributions of $\Delta_{BT}(i+1)$,

$\Delta_{CB}(i+1)$ and $\Delta_{TC}(i+1)$. The expression of $\Delta_{BT}(i+1)$ is

$$\begin{aligned}
\Delta_{BT}(i+1) &= \frac{1}{2} \cdot (\Delta_{TC}(i) + \Delta_{CB}(i)) + \frac{c[i+1]}{2} \cdot (\Delta_{TC}(i) - \Delta_{CB}(i)) \\
&\quad + \frac{c[i+1]}{2} \cdot \left(p_{i+1}^{(B)} - q_{i+1}^{(T)} - q_{i+1}^{(B)} + r_{i+1}^{(T)} \right) \\
&\quad + \frac{1}{2} \left(p_{i+1}^{(B)} - q_{i+1}^{(T)} + q_{i+1}^{(B)} - r_{i+1}^{(T)} \right) \\
&= -\frac{1}{2} \Delta_{BT}(i) + \frac{c[i+1]}{2} \cdot (\Delta_{TC}(i) - \Delta_{CB}(i)) \\
&\quad + \frac{1+c[i+1]}{2} \left(p_{i+1}^{(B)} - q_{i+1}^{(T)} \right) + \frac{1-c[i+1]}{2} \left(q_{i+1}^{(B)} - r_{i+1}^{(T)} \right).
\end{aligned}$$

Here $\Delta_{BT}(i), \Delta_{CB}(i)$ follow $\mathcal{N}(0, \sqrt{i+1}\sigma)$ and $\Delta_{TC}(i)$ follows $\mathcal{N}(0, \sqrt{2(i+1)}\sigma)$. The delay parameters $p_{i+1}^{(j)}, q_{i+1}^{(j)}, r_{i+1}^{(j)}$ follow $\mathcal{N}(\mu, \sigma)$. It can be observed that $\Delta_{BT}(i+1)$ will follow $\mathcal{N}(0, \sigma')$ where

$$\begin{aligned}
\sigma' &= \frac{1}{4} \left((i+1)\sigma^2 \right) + \frac{1}{4} \left(2(i+1)\sigma^2 + (i+1)\sigma^2 \right) + \frac{1+c[i+1]^2}{4} 2\sigma^2 \\
&= (i+2)\sigma^2.
\end{aligned}$$

Hence $\Delta_{BT}(i+1) \sim \mathcal{N}(0, \sqrt{i+2}\sigma)$. Similarly it can also be shown that $\Delta_{CB}(i+1) \sim \mathcal{N}(0, \sqrt{i+2}\sigma)$. As $\Delta_{TC}(i+1) = -(\Delta_{BT}(i+1) + \Delta_{CB}(i+1))$, $\Delta_{TC}(i+1)$ will follow $\mathcal{N}(0, \sqrt{2(i+2)}\sigma)$. Hence from the mathematical induction we get $\Delta_{BT}(n), \Delta_{CB}(n) \sim \mathcal{N}(0, \sqrt{n+1}\sigma)$ and $\Delta_{TC}(n) \sim \mathcal{N}(0, \sqrt{2(n+1)}\sigma)$ for $n \geq 0$. \square

We now consider two challenge inputs C and \tilde{C} to an n -length PA-PUF where C and \tilde{C} differ at the last bit i.e., if $C = (c[0], \dots, c[n-1])$ then $\tilde{C} = (c[0], \dots, -c[n-1])$. Let z_C and $z_{\tilde{C}}$ are the outputs from the n -length PA-PUF corresponding to challenge inputs C and \tilde{C} . We are now interested to calculate $\Pr[z_C = z_{\tilde{C}}]$ for the mentioned type of C and \tilde{C} .

Theorem 11. *If z_C and $z_{\tilde{C}}$ are the output bits from an n -length PA-PUF corresponding to two challenge inputs C and \tilde{C} , where C and \tilde{C} differ at the $(n-1)$ -th ($n \geq 1$) position then $\Pr[z_C = z_{\tilde{C}}] = \frac{1}{\pi} \tan^{-1} \frac{2\sqrt{1-\rho^2}\sqrt{n+3}\sqrt{3n+1}}{2n-2}$ with $\rho = \frac{n-1}{\sqrt{n+3}\sqrt{3n+1}}$.*

Proof. We have already observed that the output bit from an n -length PA-PUF is decided on the sign of $\Delta_{BT}(n-1)$. Thus we will look into the change of sign in $\Delta_{BT}(n-1)$ with the change of sign in $c[n-1]$ of $C = (c[0], \dots, c[n-1])$. The

expression of $\Delta_{BT}(n-1)$ is

$$\begin{aligned}
\Delta_{BT}(n-1) &= \frac{1}{2} \cdot (\Delta_{TC}(n-2) + \Delta_{CB}(n-2)) \\
&\quad + \frac{c[n-1]}{2} \cdot (\Delta_{TC}(n-2) - \Delta_{CB}(n-2)) \\
&\quad + \frac{c[n-1]}{2} \cdot \left(p_{n-1}^{(B)} - q_{n-1}^{(T)} - q_{n-1}^{(B)} + r_{n-1}^{(T)} \right) \\
&\quad + \frac{1}{2} \left(p_{n-1}^{(B)} - q_{n-1}^{(T)} + q_{n-1}^{(B)} - r_{n-1}^{(T)} \right).
\end{aligned}$$

Now with the change in sign of $c[n-1]$ the sign of $\Delta_{BT}(n-1)$ will not alter if and only if

$$\begin{aligned}
&|(\Delta_{TC}(n-2) + \Delta_{CB}(n-2)) + (p_{n-1}^{(B)} - q_{n-1}^{(T)} + q_{n-1}^{(B)} - r_{n-1}^{(T)})| \\
&> |(\Delta_{TC}(n-2) - \Delta_{CB}(n-2)) + (p_{n-1}^{(B)} - q_{n-1}^{(T)} - q_{n-1}^{(B)} + r_{n-1}^{(T)})| \\
\Rightarrow &|(\Delta_{TC}(n-2) + p_{n-1}^{(B)} - q_{n-1}^{(T)}) + (\Delta_{CB}(n-2) + q_{n-1}^{(B)} - r_{n-1}^{(T)})| \\
&> |(\Delta_{TC}(n-2) + p_{n-1}^{(B)} - q_{n-1}^{(T)}) - (\Delta_{CB}(n-2) + q_{n-1}^{(B)} - r_{n-1}^{(T)})|. \quad (5.3)
\end{aligned}$$

Let $X = \Delta_{TC}(n-2) + p_{n-1}^{(B)} - q_{n-1}^{(T)}$ and $Y = \Delta_{CB}(n-2) + q_{n-1}^{(B)} - r_{n-1}^{(T)}$. From our earlier discussion $\Delta_{TC}(n-2) \sim \mathcal{N}(0, \sqrt{2n-2}\sigma)$ and $\Delta_{CB}(n-2) \sim \mathcal{N}(0, \sqrt{n-1}\sigma)$. Now X, Y will follow $\mathcal{N}(0, \sigma_x)$ and $\mathcal{N}(0, \sigma_y)$ respectively with $\sigma_x = \sqrt{2n}\sigma$ and $\sigma_y = \sqrt{n+1}\sigma$. With this substitution we get, $\Delta_{BT}(n-1)$ will not alter sign if and only if $|X+Y| > |X-Y|$. Hence $\Pr[z_C = z_{\bar{C}}] = \Pr[|X+Y| > |X-Y|]$. Thus we need to calculate $\Pr\left[\left|\frac{X+Y}{X-Y}\right| > 1\right]$. We already know that $\Delta_{BT}(n-2), \Delta_{CB}(n-2) \sim \mathcal{N}(0, \sqrt{n-1}\sigma)$ and $\Delta_{TC}(n-2) \sim \mathcal{N}(0, \sqrt{2n-2}\sigma)$. The delay parameters $p_{n-1}^{(B)}, q_{n-1}^{(T)}, q_{n-1}^{(B)}, r_{n-1}^{(T)}$ follow $\mathcal{N}(\mu, \sigma)$. We further simplify $X+Y$ and $X-Y$.

$$\begin{aligned}
X+Y &= \Delta_{TC}(n-2) + p_{n-1}^{(B)} - q_{n-1}^{(T)} + \Delta_{CB}(n-2) + q_{n-1}^{(B)} - r_{n-1}^{(T)} \\
&= \Delta_{TC}(n-2) + \Delta_{CB}(n-2) + p_{n-1}^{(B)} - q_{n-1}^{(T)} + q_{n-1}^{(B)} - r_{n-1}^{(T)} \\
&= -\Delta_{BT}(n-2) + p_{n-1}^{(B)} - q_{n-1}^{(T)} + q_{n-1}^{(B)} - r_{n-1}^{(T)} \quad (\text{from Proposition 1}).
\end{aligned}$$

$$\begin{aligned}
X-Y &= (\Delta_{TC}(n-2) + p_{n-1}^{(B)} - q_{n-1}^{(T)}) - (\Delta_{CB}(n-2) + q_{n-1}^{(B)} - r_{n-1}^{(T)}) \\
&= \Delta_{TC}(n-2) - \Delta_{CB}(n-2) + p_{n-1}^{(B)} - q_{n-1}^{(T)} - q_{n-1}^{(B)} + r_{n-1}^{(T)}.
\end{aligned}$$

It can be easily checked that $X+Y$ will follow $\mathcal{N}(0, \sigma_{x+y})$ and $X-Y$ will follow

$\mathcal{N}(0, \sigma_{x-y})$ where $\sigma_{x+y} = \sqrt{n+3}\sigma$, $\sigma_{x-y} = \sqrt{3n+1}\sigma$. We are now interested to find the joint probability distribution of $X+Y$ and $X-Y$. To find the required joint probability distribution we first compute the $Cov(X+Y, X-Y)$. We know that $Cov(X+Y, X-Y) = \text{Variance}(X) - \text{Variance}(Y) = \sigma_x^2 - \sigma_y^2 = (n-1)\sigma^2$. The correlation (ρ) between $(X+Y)$ and $(X-Y)$ will be

$$\rho = \frac{Cov(X+Y, X-Y)}{\sigma_{x+y}\sigma_{x-y}} = \frac{n-1}{\sqrt{n+3}\sqrt{3n+1}}.$$

Let $X_1 = X+Y$, $X_2 = X-Y$, $\sigma_{x_1} = \sigma_{x+y}$ and $\sigma_{x_2} = \sigma_{x-y}$. The probability density function of X_1 and X_2 will be,

$$f_{X_1}(x_1) = \frac{1}{\sqrt{2\pi}\sigma_{X_1}} e^{-\frac{x_1^2}{2\sigma_{X_1}^2}}, \quad -\infty < x_1 < \infty,$$

$$f_{X_2}(x_2) = \frac{1}{\sqrt{2\pi}\sigma_{X_2}} e^{-\frac{x_2^2}{2\sigma_{X_2}^2}}, \quad -\infty < x_2 < \infty.$$

We need to calculate the probability $\Pr\left[\left|\frac{X+Y}{X-Y}\right| > 1\right]$ which is basically $\Pr\left[\left|\frac{X_1}{X_2}\right| > 1\right]$. As X_1, X_2 are not mutually independent random variables, the joint probability distribution of X_1, X_2 will be,

$$f_{X_1, X_2}(x_1, x_2) = \frac{1}{2\pi\sigma_{x_1}\sigma_{x_2}\sqrt{1-\rho^2}} e^{-\frac{1}{2} \frac{\frac{x_1^2}{\sigma_{x_1}^2} - 2\rho \frac{x_1}{\sigma_{x_1}} \frac{x_2}{\sigma_{x_2}} + \frac{x_2^2}{\sigma_{x_2}^2}}{1-\rho^2}}, \quad -\infty < x_1, x_2 < \infty.$$

Let $Y_1 = \frac{X_1}{X_2}$ and $Y_2 = X_2$. Then $X_1 = Y_1 Y_2$ and $X_2 = Y_2$ and $-\infty < y_1, y_2 < \infty$ if $-\infty < x_1, x_2 < \infty$. We further derive the joint probability distribution of Y_1, Y_2 .

$$f_{Y_1, Y_2}(y_1, y_2) = f_{X_1, X_2}(y_1 y_2, y_2) |J|, \quad \text{where } |J| = \begin{vmatrix} y_2 & y_1 \\ 0 & 1 \end{vmatrix}$$

$$= \frac{1}{2\pi\sigma_{x_1}\sigma_{x_2}\sqrt{1-\rho^2}} e^{-\frac{1}{2} \frac{\frac{y_1^2 y_2^2}{\sigma_{x_1}^2} - 2\rho \frac{y_1 y_2}{\sigma_{x_1}\sigma_{x_2}} + \frac{y_2^2}{\sigma_{x_2}^2}}{1-\rho^2}} y_2; \quad -\infty < y_1, y_2 < \infty.$$

As we are interested in calculating $\Pr\left[\left|\frac{X_1}{X_2}\right| > 1\right] = \Pr[|Y_1| > 1]$, we need to derive

the distribution of Y_1 . Below we calculate the probability distribution of Y_1 .

$$\begin{aligned}
f_{Y_1}(y_1) &= \int_{-\infty}^{\infty} f_{Y_1, Y_2}(y_1, y_2) dy_2 \\
&= \int_{-\infty}^{\infty} \frac{1}{2\pi\sigma_{x_1}\sigma_{x_2}\sqrt{1-\rho^2}} e^{-\frac{1}{2} \frac{\frac{y_1^2 y_2^2}{\sigma_{x_1}^2} - 2\rho \frac{y_1 y_2}{\sigma_{x_1}\sigma_{x_2}} + \frac{y_2^2}{\sigma_{x_2}^2}}{1-\rho^2}} y_2 dy_2 \\
&= \int_{-\infty}^{\infty} \frac{1}{2\pi\sigma_{x_1}\sigma_{x_2}\sqrt{1-\rho^2}} e^{-\frac{1}{2} \frac{\left(\frac{y_1^2}{\sigma_{x_1}^2} - 2\rho \frac{y_1}{\sigma_{x_1}\sigma_{x_2}} + \frac{1}{\sigma_{x_2}^2}\right) y_2^2}{1-\rho^2}} y_2 dy_2 \\
&= \frac{1}{\pi\sigma_{x_1}\sigma_{x_2}\sqrt{1-\rho^2}} \frac{1-\rho^2}{\frac{y_1^2}{\sigma_{x_1}^2} - 2\rho \frac{y_1}{\sigma_{x_1}\sigma_{x_2}} + \frac{1}{\sigma_{x_2}^2}} \\
&= \frac{\sqrt{1-\rho^2}}{\pi} \frac{\frac{\sigma_{x_1}}{\sigma_{x_2}}}{y_1 - 2y_1 \frac{\rho\sigma_{x_1}}{\sigma_{x_2}} + \frac{\sigma_{x_1}^2}{\sigma_{x_2}^2}} \\
&= \frac{1}{\pi} \frac{\frac{\sqrt{1-\rho^2}\sigma_{x_1}}{\sigma_{x_2}}}{\left(y_1 - \frac{\rho\sigma_{x_1}}{\sigma_{x_2}}\right)^2 + \frac{(1-\rho^2)\sigma_{x_1}^2}{\sigma_{x_2}^2}} \\
&= \frac{1}{\pi} \frac{a}{(y_1 - b)^2 + a^2}, \text{ where } a = \frac{\sqrt{1-\rho^2}\sigma_{x_1}}{\sigma_{x_2}}, b = \frac{\rho\sigma_{x_1}}{\sigma_{x_2}}.
\end{aligned}$$

We need to find $\Pr\left[|Y_1| > 1\right]$. Below, we derive this required probability.

$$\begin{aligned}
\Pr\left[|Y_1| > 1\right] &= 1 - \Pr\left[|Y_1| < 1\right] \\
&= 1 - \left| \int_{-1}^1 \frac{1}{\pi} \frac{a}{(y_1 - b)^2 + a^2} dy_1 \right| \\
&= 1 - \left| \int_{-1-b}^{1-b} \frac{1}{\pi} \frac{a}{z^2 + a^2} dz \right|, \text{ substituting } z = y_1 - b \\
&= 1 - \left| \frac{1}{\pi} \tan^{-1} \left(\frac{y_1}{a} \right) \Big|_{-1-b}^{1-b} \right| \\
&= 1 - \left| \frac{1}{\pi} \tan^{-1} \frac{1-b}{a} + \tan^{-1} \frac{1+b}{a} \right| \\
&= 1 - \left| \frac{1}{\pi} \tan^{-1} \frac{\frac{1-b}{a} + \frac{1+b}{a}}{1 - \frac{1-b}{a} \frac{1+b}{a}} \right| \\
&= 1 - \left| \frac{1}{\pi} \tan^{-1} \frac{2a}{a^2 + b^2 - 1} \right|
\end{aligned}$$

Here $a = \frac{\sqrt{1-\rho^2}\sigma_{x_1}}{\sigma_{x_2}}$ and $b = \frac{\rho\sigma_{x_1}}{\sigma_{x_2}}$. Thus

$$\begin{aligned} a^2 + b^2 - 1 &= (1 - \rho^2) \frac{\sigma_{x_1}^2}{\sigma_{x_2}^2} + \frac{\rho^2 \sigma_{x_1}^2}{\sigma_{x_2}^2} - 1 \\ &= \frac{\sigma_{x_1}^2}{\sigma_{x_2}^2} - 1. \end{aligned}$$

$$\begin{aligned} \Pr\left[\left|Y_1\right| > 1\right] &= 1 - \left|\frac{1}{\pi} \tan^{-1} \frac{2a}{a^2 + b^2 - 1}\right| \\ &= 1 - \left|\frac{1}{\pi} \tan^{-1} \frac{2\sqrt{1-\rho^2}\sigma_{x_1}}{\frac{\sigma_{x_1}^2}{\sigma_{x_2}^2} - 1}\right| \\ &= 1 - \left|\frac{1}{\pi} \tan^{-1} \frac{2\sqrt{1-\rho^2}\sigma_{x_1}\sigma_{x_2}}{\sigma_{x_1}^2 - \sigma_{x_2}^2}\right| \\ &= 1 - \left|\frac{1}{\pi} \tan^{-1} \frac{2\sqrt{1-\rho^2}\sqrt{n+3}\sqrt{3n+1}}{(n+3) - (3n+1)}\right| \\ &= 1 - \left|\frac{1}{\pi} \tan^{-1} \frac{2\sqrt{1-\rho^2}\sqrt{n+3}\sqrt{3n+1}}{-2n+2}\right| \\ &= 1 - \frac{1}{\pi} \left(\pi - \tan^{-1} \frac{2\sqrt{1-\rho^2}\sqrt{n+3}\sqrt{3n+1}}{2n-2}\right) \\ &= \frac{1}{\pi} \tan^{-1} \frac{2\sqrt{1-\rho^2}\sqrt{n+3}\sqrt{3n+1}}{2n-2}. \end{aligned}$$

Hence $\Pr[z_C = z_{\tilde{C}}] = \frac{1}{\pi} \tan^{-1} \frac{2\sqrt{1-\rho^2}\sqrt{n+3}\sqrt{3n+1}}{2n-2}$ for $C = (c[0], \dots, c[n-1])$ and $\tilde{C} = (c[0], \dots, -c[n-1])$, where $\rho = \frac{n-1}{\sqrt{n+3}\sqrt{3n+1}}$. \square

We perform experiments using 2^{10} many random PA-PUFs to verify the derived probability. Comparison between experimental estimation and theoretically estimation is given in Table 5.3.

The next question that comes to our mind is that if the $(n-2)$ -th (i.e., second last) position of the challenge input is altered then in what probability two outputs will match? We will discuss this probability in the next theorem.

Theorem 12. *Let C and \tilde{C} be two challenge inputs of an n -length PA-PUF where $C = (c[0], \dots, c[n-2], c[n-1])$ and $\tilde{C} = (c[0], \dots, -c[n-2], c[n-1])$. If z_C and $z_{\tilde{C}}$*

n	Experimental $\Pr[z_C = z_{\bar{C}}]$	Theoretical $\Pr[z_C = z_{\bar{C}}]$
5	0.3734	0.3848
6	0.3714	0.3750
7	0.3624	0.3673
8	0.3594	0.3611
9	0.3558	0.3561
10	0.3530	0.3519

Table 5.3: Experimental validation of Theorem 11

be the respective output bits from the PA-PUF, then

$$\Pr[z_C = z_{\bar{C}}] = \frac{1}{2} \left(\frac{1}{\pi} \tan^{-1} \frac{2\sqrt{1-\rho^2}\sigma_1\sigma_2}{\sigma_2^2 - \sigma_1^2} \Big|_{c[n-1]=-1} + \frac{1}{\pi} \tan^{-1} \frac{2\sigma_1\sigma_2}{\sigma_1^2 - \sigma_2^2} \Big|_{c[n-1]=1} \right)$$

Here $A_{n-1} = \frac{1+c[n-1]}{2}$, $B_{n-1} = \frac{1-c[n-1]}{2}$, $\sigma_1 = \sigma^2 \left((n-2)(1 + A_{n-1}^2) + 12 \right)$, $\sigma_2 = \sigma^2 \left((n-2)(2 + B_{n-1}^2) + 4 \right)$, $\rho = \frac{-^{(n-2)}B_{n-1}^2}{\sqrt{\left((n-2)(1+A_{n-1}^2)+12 \right) \left((n-2)(2+B_{n-1}^2)+4 \right)}}$ and σ is the standard deviation of the distribution of the delay parameters.

Proof. To prove this we consider the expression of $\Delta_{BT}(n-1)$.

$$\begin{aligned} & \Delta_{BT}(n-1) \\ &= \frac{1}{2} \cdot (\Delta_{TC}(n-2) + \Delta_{CB}(n-2)) + \frac{c[n-1]}{2} \cdot (\Delta_{TC}(n-2) - \Delta_{CB}(n-2)) \\ & \quad + \frac{c[n-1]}{2} \cdot \left(p_{n-1}^{(B)} - q_{n-1}^{(T)} - q_{n-1}^{(B)} + r_{n-1}^{(T)} \right) \\ & \quad + \frac{1}{2} \left(p_{n-1}^{(B)} - q_{n-1}^{(T)} + q_{n-1}^{(B)} - r_{n-1}^{(T)} \right) \\ &= \frac{1+c[n-1]}{2} \Delta_{TC}(n-2) + \frac{1-c[n-1]}{2} \Delta_{CB}(n-2) \\ & \quad + \frac{1+c[n-1]}{2} (p_{n-1}^{(B)} - q_{n-1}^{(T)}) + \frac{1-c[n-1]}{2} (q_{n-1}^{(B)} - r_{n-1}^{(T)}). \end{aligned}$$

Let $A_{n-1} = \frac{1+c[n-1]}{2}$, $B_{n-1} = \frac{1-c[n-1]}{2}$, $D_{n-1} = p_{n-1}^{(B)} - q_{n-1}^{(T)}$ and $E_{n-1} = q_{n-1}^{(B)} - r_{n-1}^{(T)}$. The expression of $\Delta_{BT}(n-1)$ now reduces to $\Delta_{BT}(n-1) = A_{n-1}\Delta_{TC}(n-2) + B_{n-1}\Delta_{CB}(n-2) + A_{n-1}D_{n-1} + B_{n-1}E_{n-1}$.

$$\Delta_{BT}(n-1) = A_{n-1}\Delta_{TC}(n-2) + B_{n-1}\Delta_{CB}(n-2) + A_{n-1}D_{n-1} + B_{n-1}E_{n-1}$$

$$\begin{aligned}
&= A_{n-1} \left\{ \frac{1+c[n-2]}{2} \Delta_{CB}(n-3) + \frac{1-c[n-2]}{2} \Delta_{BT}(n-3) \right. \\
&\quad \left. + \frac{1+c[n-2]}{2} (q_{n-2}^{(T)} - r_{n-2}^{(C)}) + \frac{1-c[n-2]}{2} (r_{n-2}^{(T)} - p_{n-2}^{(C)}) \right\} \\
&\quad + B_{n-1} \left\{ \frac{1+c[n-2]}{2} \Delta_{BT}(n-3) + \frac{1-c[n-2]}{2} \Delta_{TC}(n-3) \right. \\
&\quad \left. + \frac{1+c[n-2]}{2} (r_{n-2}^{(C)} - p_{n-2}^{(B)}) + \frac{1-c[n-2]}{2} (p_{n-2}^{(C)} - q_{n-2}^{(B)}) \right\} \\
&\quad + A_{n-1} D_{n-1} + B_{n-1} D_{n-1} \\
&= \frac{A_{n-1}}{2} \left\{ \Delta_{CB}(n-3) + \Delta_{BT}(n-3) + (q_{n-2}^{(T)} - r_{n-2}^{(C)} + (r_{n-2}^{(T)} - p_{n-2}^{(C)})) \right\} \\
&\quad + \frac{A_{n-1}c[n-2]}{2} \left\{ \Delta_{CB}(n-3) - \Delta_{BT}(n-3) + (q_{n-2}^{(T)} - r_{n-2}^{(C)}) - (r_{n-2}^{(T)} - p_{n-2}^{(C)}) \right\} \\
&\quad + \frac{B_{n-1}}{2} \left\{ \Delta_{BT}(n-3) + \Delta_{TC}(n-3) + (r_{n-2}^{(C)} - p_{n-2}^{(B)} + (p_{n-2}^{(C)} - q_{n-2}^{(B)})) \right\} \\
&\quad + \frac{B_{n-1}c[n-2]}{2} \left\{ \Delta_{BT}(n-3) - \Delta_{TC}(n-3) + (r_{n-2}^{(C)} - p_{n-2}^{(B)}) - (p_{n-2}^{(T)} - q_{n-2}^{(C)}) \right\} \\
&\quad + (A_{n-1}D_{n-1} + B_{n-1}D_{n-1}) \\
&= \frac{c[n-2]}{2} \left\{ A_{n-1} \left(\Delta_{CB}(n-3) - \Delta_{BT}(n-3) + (q_{n-2}^{(T)} - r_{n-2}^{(C)}) - (r_{n-2}^{(T)} - p_{n-2}^{(C)}) \right) \right. \\
&\quad \left. + B_{n-1} \left(\Delta_{BT}(n-3) - \Delta_{TC}(n-3) + (r_{n-2}^{(C)} - p_{n-2}^{(B)}) - (p_{n-2}^{(C)} - q_{n-2}^{(B)}) \right) \right\} \\
&\quad + \frac{1}{2} \left\{ A_{n-1} \left(\Delta_{CB}(n-3) + \Delta_{BT}(n-3) + (q_{n-2}^{(T)} - r_{n-2}^{(C)}) + (r_{n-2}^{(T)} - p_{n-2}^{(C)}) \right) \right. \\
&\quad \left. + B_{n-1} \left(\Delta_{BT}(n-3) + \Delta_{TC}(n-3) + (r_{n-2}^{(C)} - p_{n-2}^{(B)}) + (p_{n-2}^{(C)} - q_{n-2}^{(B)}) \right) \right\} \\
&\quad + (A_{n-1}D_{n-1} + B_{n-1}D_{n-1}).
\end{aligned}$$

Let $X = A_{n-1} \left(\Delta_{CB}(n-3) + (q_{n-2}^{(T)} - r_{n-2}^{(C)}) \right) + B_{n-1} \left(\Delta_{BT}(n-3) + (r_{n-2}^{(C)} - p_{n-2}^{(B)}) \right)$,
 $Y = A_{n-1} \left(\Delta_{BT}(n-3) + (r_{n-2}^{(T)} - p_{n-2}^{(C)}) \right) + B_{n-1} \left(\Delta_{TC}(n-3) + (p_{n-2}^{(C)} - q_{n-2}^{(B)}) \right)$ and
 $Z = (A_{n-1}D_{n-1} + B_{n-1}E_{n-1})$. With this substitution we get

$$\Delta_{BT}(n-1) = \frac{c[n-2]}{2}(X - Y) + \frac{1}{2}(X + Y) + Z.$$

Now with the change in sign of $c[n-2]$, $\Delta_{BT}(n-1)$ will have same sign if and only if $|X + Y + 2Z| > |X - Y|$. It can be noticed that $X \sim \mathcal{N}(0, \sigma_X)$, $Y \sim \mathcal{N}(0, \sigma_Y)$ and $Z \sim \mathcal{N}(0, \sigma_Z)$ where

$$\begin{aligned}
\sigma_X^2 &= A_{n-1}^2 \left((n-2)\sigma^2 + 2\sigma^2 \right) + B_{n-1}^2 \left((n-2)\sigma^2 + 2\sigma^2 \right) \\
&= n\sigma^2, \\
\sigma_Y^2 &= A_{n-1}^2 \left((n-2)\sigma^2 + 2\sigma^2 \right) + B_{n-1}^2 \left(2(n-2)\sigma^2 + 2\sigma^2 \right) \\
&= \sigma^2 \left(n + (n-2)B_{n-1}^2 \right), \\
\sigma_Z^2 &= A_{n-1}^2 2\sigma^2 + B_{n-1}^2 2\sigma^2 = 2\sigma^2.
\end{aligned}$$

We are now interested in calculating the probability distributions of $(X + Y + 2Z)$ and $(X - Y)$. Here $(X + Y + 2Z) \sim \mathcal{N}(0, \sigma_1)$ and $(X - Y) \sim \mathcal{N}(0, \sigma_2)$. Here we need to calculate σ_1 and σ_2 .

$$\begin{aligned}
X + Y + 2Z &= A_{n-1} \left(\Delta_{CB}(n-3) + (q_{n-2}^{(T)} - r_{n-2}^{(C)}) \right) \\
&\quad + B_{n-1} \left(\Delta_{BT}(n-3) + (r_{n-2}^{(C)} - p_{n-2}^{(B)}) \right) \\
&\quad + A_{n-1} \left(\Delta_{BT}(n-3) + (r_{n-2}^{(T)} - p_{n-2}^{(C)}) \right) \\
&\quad + B_{n-1} \left(\Delta_{TC}(n-3) + (p_{n-2}^{(C)} - q_{n-2}^{(B)}) \right) \\
&\quad + 2(A_{n-1}D_{n-1} + B_{n-1}E_{n-1}) \\
&= A_{n-1}(-\Delta_{TC}(n-3)) + B_{n-1}(-\Delta_{CB}(n-3)) \\
&\quad + A_{n-1}(q_{n-2}^{(T)} - r_{n-2}^{(C)} + r_{n-2}^{(T)} - p_{n-2}^{(C)}) \\
&\quad + B_{n-1}(r_{n-2}^{(C)} - p_{n-2}^{(B)} + p_{n-2}^{(C)} - q_{n-2}^{(B)}) \\
&\quad + 2(A_{n-1}D_{n-1} + B_{n-1}E_{n-1}).
\end{aligned}$$

Hence

$$\begin{aligned}
\sigma_1^2 &= A_{n-1}^2 2(n-2)\sigma^2 + B_{n-1}^2 (n-2)\sigma^2 + A_{n-1}^2 4\sigma^2 + B_{n-1}^2 4\sigma^2 \\
&\quad + 4(A_{n-1}^2 2\sigma^2 + B_{n-1}^2 2\sigma^2) \\
&= \sigma^2 \left((n-2)(2A_{n-1}^2 + B_{n-1}^2) + 4 + 8 \right) \\
&= \sigma^2 \left((n-2)(1 + A_{n-1}^2) + 12 \right).
\end{aligned}$$

Similarly,

$$\begin{aligned}
X - Y &= A_{n-1} \left(\Delta_{CB}(n-3) + (q_{n-2}^{(T)} - r_{n-2}^{(C)}) \right) \\
&\quad + B_{n-1} \left(\Delta_{BT}(n-3) + (r_{n-2}^{(C)} - p_{n-2}^{(B)}) \right) \\
&\quad - \left(A_{n-1} \left(\Delta_{BT}(n-3) + (r_{n-2}^{(T)} - p_{n-2}^{(C)}) \right) \right. \\
&\quad \left. + B_{n-1} \left(\Delta_{TC}(n-3) + (p_{n-2}^{(C)} - q_{n-2}^{(B)}) \right) \right) \\
&= A_{n-1} \left((\Delta_{CB}(n-3) - \Delta_{BT}(n-3)) \right. \\
&\quad \left. + (q_{n-2}^{(T)} - r_{n-2}^{(C)}) - (r_{n-2}^{(T)} - p_{n-2}^{(C)}) \right) \\
&\quad + B_{n-1} \left((\Delta_{BT}(n-3) - \Delta_{TC}(n-3)) \right)
\end{aligned}$$

$$+(r_{n-2}^{(C)} - p_{n-2}^{(B)}) - (p_{n-2}^{(C)} - q_{n-2}^{(B)})$$

Here $(X - Y) \sim \mathcal{N}(0, \sigma_2)$ with

$$\begin{aligned}\sigma_2^2 &= A_{n-1}^2(2(n-2)\sigma^2 + 4\sigma^2) + B_{n-1}^2(3(n-2)\sigma^2 + 4\sigma^2) \\ &= \sigma^2((n-2)(2A_{n-1}^2 + 3B_{n-1}^2) + 4) \\ &= \sigma^2((n-2)(2 + B_{n-1}^2) + 4).\end{aligned}$$

Now we would like to check the $Cov(X + Y + 2Z, X - Y)$ as we are interested to find the joint probability distribution of $X + Y + 2Z$ and $X - Y$ for finding $\Pr[|X + Y + 2Z| > |X - Y|]$.

$$\begin{aligned}&Cov(X + Y + 2Z, X - Y) \\ &= E((X + Y + 2Z)(X - Y)) - E(X + Y + 2Z)E(X - Y) \\ &= E((X^2 - Y^2) + Z(X - Y)) - (E(X) + E(Y) + 2E(Z))(E(X) - E(Y)) \\ &= E(X^2) - E(Y^2) + E(Z(X - Y)) \text{ here } E(X) = E(Y) = E(Z) = 0 \\ &= E(X^2) - E(Y^2) \text{ as } Z \text{ is independent from } (X - Y) \\ &= \sigma_x^2 - \sigma_y^2 \\ &= n\sigma^2 - \sigma^2(n + (n-2)B_{n-1}^2) \\ &= -(n-2)B_{n-1}^2\sigma^2.\end{aligned}$$

From the above expression of $Cov(X + Y + 2Z, X - Y)$ it can be noticed that $Cov(X + Y + 2Z, X - Y) = 0$ if $B_{n-1} = 0$ (i.e., if $c[n-1] = 1$) and $n = 2$. The correlation (ρ) between $X + Y + 2Z$ and $X - Y$ will be,

$$\begin{aligned}\rho &= \frac{Cov(X + Y + 2Z, X - Y)}{\sigma_1\sigma_2} \\ &= \frac{-(n-2)B_{n-1}^2}{\sqrt{((n-2)(1 + A_{n-1}^2) + 12)((n-2)(2 + B_{n-1}^2) + 4)}}.\end{aligned}$$

We do have two cases one is when $\rho \neq 0$ i.e., $c[n-1] = -1, n \neq 2$ (Case I) and when $\rho = 0$ i.e., $c[n-1] = 1$ (Case II).

Case I. Under this case $c[n-1] = -1, n \neq 2$ i.e., $A_{n-1} = 0$ and $B_{n-1} = 1$. Here we get the non-zero value for ρ , where $\rho = \frac{-(n-2)}{\sqrt{(n+10)(3n-2)}} \neq 0$. From the proof of Theorem 11 we can directly say that $\Pr[z_C = z_{\tilde{C}}] = 1 - \frac{1}{\pi} \left| \tan^{-1} \frac{2\sqrt{1-\rho^2}\sigma_1\sigma_2}{\sigma_1^2 - \sigma_2^2} \right|$.

$$\begin{aligned}
P_1 &= \Pr[z_C = z_{\tilde{C}}] \\
&= 1 - \frac{1}{\pi} \left| \tan^{-1} \frac{2\sqrt{1-\rho^2}\sigma_1\sigma_2}{\sigma_1^2 - \sigma_2^2} \right| \\
&= 1 - \frac{1}{\pi} \left(\pi - \tan^{-1} \frac{2\sqrt{1-\rho^2}\sigma_1\sigma_2}{\sigma_2^2 - \sigma_1^2} \right) \\
&= \frac{1}{\pi} \tan^{-1} \frac{2\sqrt{1-\rho^2}\sigma_1\sigma_2}{\sigma_2^2 - \sigma_1^2}, \text{ when } (\sigma_2^2 - \sigma_1^2) > 0.
\end{aligned}$$

Case II. Under this case $c[n-1] = 1$ i.e., $A_{n-1} = 1$ and $B_{n-1} = 0$. For this case we get $\rho = 0$ i.e., $(X+Y+2Z)$ is uncorrelated with $(X-Y)$. If $Y_1 = \frac{X+Y+2Z}{X-Y}$ then from the proof of Theorem 11 we get $a = \frac{\sigma_1}{\sigma_2}$ and $b = 0$.

$$\begin{aligned}
P_2 &= \Pr\left[|Y_1| > 1\right] \\
&= 1 - \Pr\left[|Y_1| < 1\right] \\
&= 1 - \left| \int_{-1}^1 \frac{1}{\pi} \frac{a}{y_1^2 + a^2} dy_1 \right| \\
&= 1 - \left| \frac{1}{\pi} \tan^{-1} \left(\frac{y_1}{a} \right) \Big|_{-1}^1 \right| \\
&= 1 - \left| \frac{1}{\pi} \left(\tan^{-1} \frac{1}{a} - \tan^{-1} \frac{-1}{a} \right) \right| \\
&= 1 - \left| \frac{1}{\pi} \left(\tan^{-1} \frac{1}{a} - \pi + \tan^{-1} \frac{1}{a} \right) \right| \\
&= \frac{1}{\pi} \left(\tan^{-1} \frac{1}{a} + \tan^{-1} \frac{1}{a} \right) \\
&= \frac{1}{\pi} \tan^{-1} \frac{\frac{1}{a} + \frac{1}{a}}{1 - \frac{1}{a^2}} \\
&= \frac{1}{\pi} \tan^{-1} \frac{2a}{1 - a^2} \\
&= \frac{1}{\pi} \tan^{-1} \frac{2\sigma_1\sigma_2}{\sigma_1^2 - \sigma_2^2}, \text{ note that } \sigma_1^2 - \sigma_2^2 > 0.
\end{aligned}$$

It can be noticed that P_1 is the probability of the event $(z_C = z_{\tilde{C}})$ given $c[n-1] = -1$ and P_2 is the probability of the event $(z_C = z_{\tilde{C}})$ given $c[n-1] = 1$, where $C = (c[0], \dots, c[n-2], c[n-1])$ and $\tilde{C} = (c[0], \dots, c[n-2], c[n-1])$. Thus the required

probability will be,

$$\begin{aligned}
& \Pr[z_C = z_{\tilde{C}}] \\
&= P_1 \cdot \Pr[c[n-1] = -1] + P_2 \cdot \Pr[c[n-1] = 1] \\
&= \frac{1}{2}(P_1 + P_2) \\
&= \frac{1}{2} \left(\frac{1}{\pi} \tan^{-1} \frac{2\sqrt{1-\rho^2}\sigma_1\sigma_2}{\sigma_2^2 - \sigma_1^2} \Big|_{c[n-1]=-1} + \frac{1}{\pi} \tan^{-1} \frac{2\sigma_1\sigma_2}{\sigma_1^2 - \sigma_2^2} \Big|_{c[n-1]=1} \right).
\end{aligned}$$

□

We perform experiments using 2^{10} many random PA-PUF to validate the theoretical estimation derived in Theorem 12. The comparison between theoretical estimation and experimental estimation is given in Table 5.4. From the results of Theorem 11

n	Experimental $\Pr[z_C = z_{\tilde{C}}]$	Theoretical $\Pr[z_C = z_{\tilde{C}}]$
6	0.4464	0.4596
7	0.4285	0.4549
8	0.4223	0.4511
9	0.4089	0.4477
10	0.4093	0.4449
11	0.4002	0.4424

Table 5.4: Experimental validation of Theorem 12

and 12 it can be noticed that $Pr[z_C = z_{\tilde{C}}]$ for PA-PUF differs significantly from the similar probability for classical PUF obtained by Siddhanti et al. [SBC⁺19b]. Numerical data clearly indicates that probabilities are improved for PA-PUF if the challenge inputs C and \tilde{C} differ at $(n-1)$ -th or $(n-2)$ -th position. The derivation of the theoretical formula of the probability $Pr[z_C = z_{\tilde{C}}]$ becomes more complex when C and \tilde{C} differ at other location except $(n-1)$ or $(n-2)$. We have performed experiments to compare the probability $Pr[z_C = z_{\tilde{C}}]$ for PUF and PA-PUF of 11 length, where C and \tilde{C} differ at i -th location for $i = 0, \dots, 10$. The experimental observation is presented in Table 5.5. From the experimental analysis presented in Table 5.5 and Figure 5-3 it can be noticed that the $Pr[z_C = z_{\tilde{C}}]$ has improved in PA-PUF over the classical PUF for almost all possible flip position i . We believe that deriving the theoretical formula of $Pr[z_C = z_{\tilde{C}}]$ for all i is quite challenging thus we are keeping it open.

i	$\Pr[z_C = z_{\tilde{C}}]_{\text{PUF}}$	$\Pr[z_C = z_{\tilde{C}}]_{\text{PA-PUF}}$
0	0.8696	0.8667
1	0.7564	0.7843
2	0.6787	0.7230
3	0.6171	0.6700
4	0.5514	0.6166
5	0.4993	0.5737
6	0.4456	0.5337
7	0.3837	0.4811
8	0.3207	0.4422
9	0.2476	0.3940
10	0.1360	0.3505

Table 5.5: Comparison of $\Pr[z_C = z_{\tilde{C}}]$ between PUF and PA-PUF of length 11.

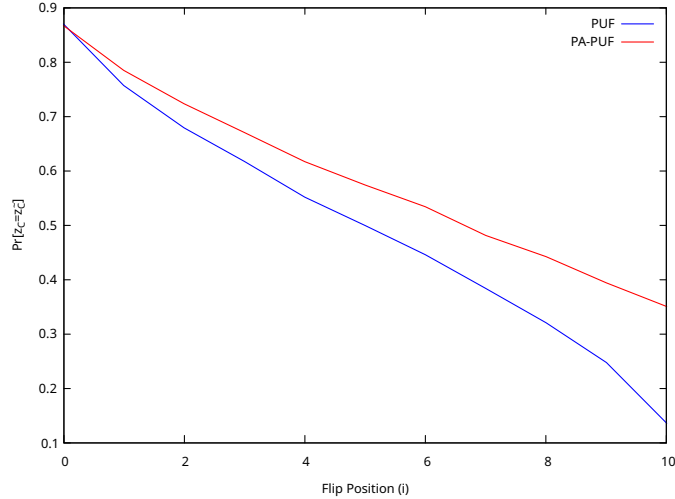


Figure 5-3: Comparison of $\Pr[z_C = z_{\tilde{C}}]$ between PUF and PA-PUF for different flip locations.

We have performed a detailed hardware-based experimental analysis on PA-PUF to compare its cryptographic properties with the existing models of PUFs. Our experimental observations are described in the following section.

5.5 Experimental Results and Analysis

The PA-PUF is designed in Verilog HDL and implemented on Nexys4 DDR FPGA. The PUF is designed with a 16-bit challenge and responses are of lengths 32, 64

and 128-bits in length. The generated responses from the PUF are collected through UART on the board. The performance of the PUF can be validated using metrics such as inter-chip Hamming distance and intra-chip Hamming distance to validate the performance of the responses generated in the PUF. There exist various metrics of PUF such as uniformity, uniqueness, bit-aliasing, and reliability.

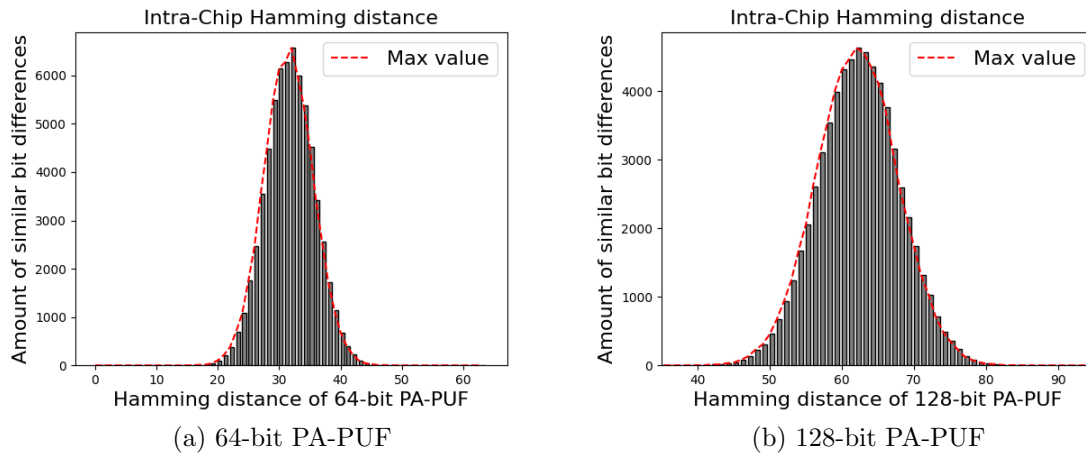


Figure 5-4: Intra-Chip Hamming Distance of the PA-PUF (a) 64-bit Response (b)128-bit Response.

5.5.1 Intra-Chip Hamming Distance

In general, a PUF should be able to generate all possible combinations of the outputs in the response. Ideally, an n -bit challenge should have 2^n combinations in the response. Out of the generated responses, the responses with uniform amounts of 0's and 1's are preferred as the key for cryptographic applications for better security. Usually, PUF will be tested for strict avalanche criterion (SAC) property, when a single bit flip is there in the challenge of the PUF 50% of the response bits should also change. The Hamming distance between two consecutive responses is used to estimate the strict avalanche property of the PUF. The experiment is conducted using a 16-bit challenge, a total of 2^{16} combinations, where two consecutive challenges differ in one-bit position. This can be observed using the intra-chip Hamming distance as shown in Figure 5-4. Figure 5-4a and 5-4b shows the intra-chip Hamming distance of the PA-PUF for the responses of 64-bit and 128-bit respectively. Ideally, the curve should be a Gaussian curve with a peak at half of its response length, indicating the

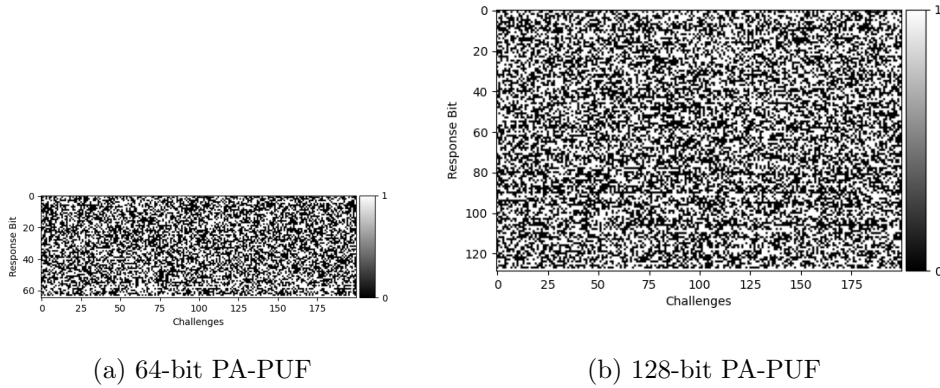


Figure 5-5: Responses of the PA-PUF to show uniformity (a)64-bit PA-PUF (b)128-bit PA-PUF.

maximum number of responses changing by 50% of the bits. For the 64-bit response, the Gaussian curve peak is at 32 while the 128-bit response has a peak at 63. This shows with a single bit change in the challenge, the responses were changing by 50%.

Intra-chip Hamming distance is used as a key to estimate uniformity and reliability.

5.5.2 Uniformity

Uniformity of the PUF is a metric which describes how uniformly 1's and 0's are distributed in the given response space. The response of the PUF should have an equal number of 1's and 0's in the response. Hence, the response can be used in cryptographic applications such as authentication and encryption resulting in better security. Figure 5-5 shows the response space for 64-bit and 128-bit responses of the PA-PUF. Figure 5-6 shows the distribution of 0's and 1's in the PUF response with the 50% probability line indicating the ideal value. It is evident to note that each response has a uniform distribution of 0's and 1's. The PA-PUF has a uniformity of 49.45% for the 128-bit response, whereas the ideal value is 50%.

5.5.3 Reliability

Electronic circuits are often prone to noise and variations in the process, voltage, and temperature result the variations in the delay, voltage, and current drawn by the circuit. Similarly, the PUF response has some influence of noise over time and

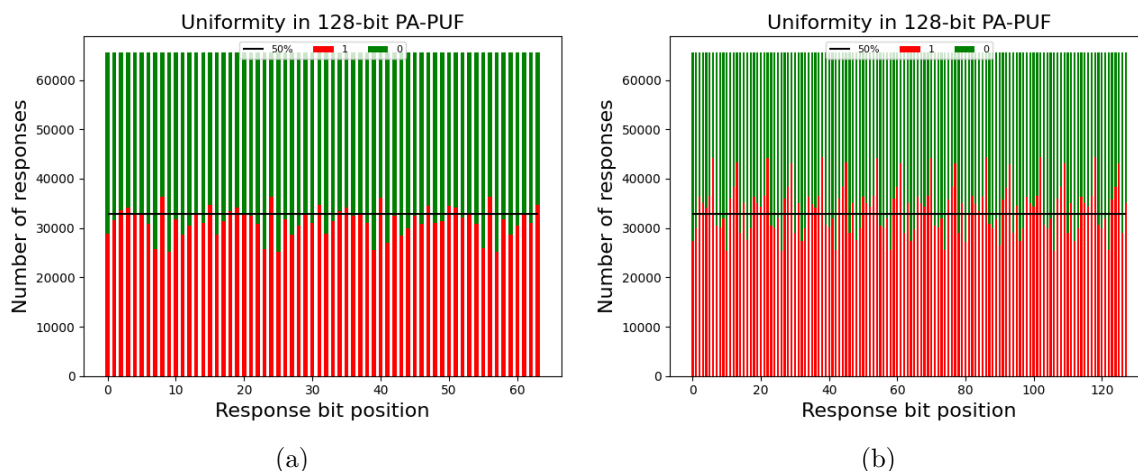


Figure 5-6: Uniformity of the PA-PUF (a) 64-bit Response (b)128-bit Response.

generates a noisy response. Hence, the reliability of the PUF is estimated to assess the stability of the PUF response. Figures 5-7 shows the responses of the PUF for the same challenge and it is evident that only a few bits have noise by toggling from 0 to 1 or 1 to 0. The errors that occurred in the PUF response can be corrected using some error correction codes such as BCH error correction codes. The reliability of the PUF can be estimated using the Hamming distance between the responses when the same challenge is given to the PUF. The Hamming distance plots shown in Figure 5-8 shows the maximum number of errors that are occurred in the PA-PUF. In particular, Figure 5-8a shows the reliability plot of the 64-bit response, where a total of seven bits were changed out of the 64-bit response. Whereas for a 128-bit response as shown in Figure 5-8b, a total of 13 bits were influenced by noise. The reliability of the PA-PUF is 98% and 95% for 64-bit and 128-bit responses, respectively. However, the errors can be corrected using BCH error correction codes leading to 100% reliable responses as shown in Figure 5-9, which shows the stability of the PUF.

5.5.4 Uniqueness

The response of the PUF should be unique to the hardware, even by repeating the experiment on similar hardware with similar constraints the response should be different, resulting in a Hamming distance of 50%. Inter-chip Hamming distance is used to assess the uniqueness of the PUF. To estimate the uniqueness, the challenge-response pairs are recorded from two (or more) FPGA boards of the same configuration. Ide-

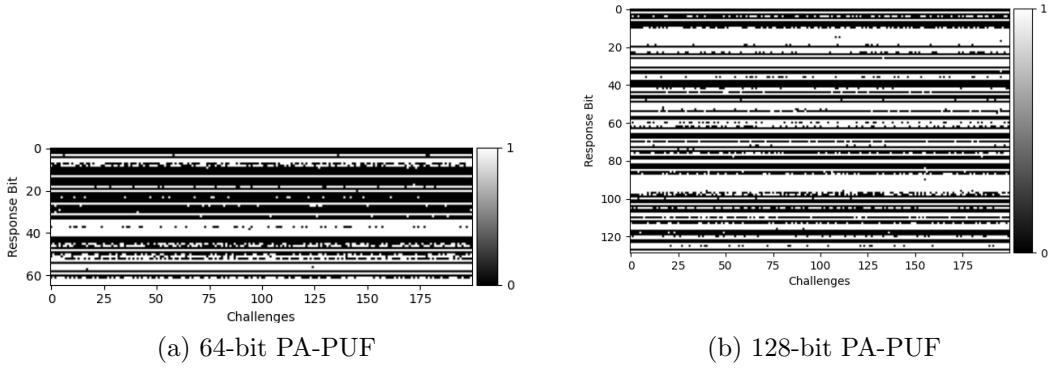


Figure 5-7: Response of the PA-PUF to show reliability (a) 64-bit Response (b)128-bit Response.

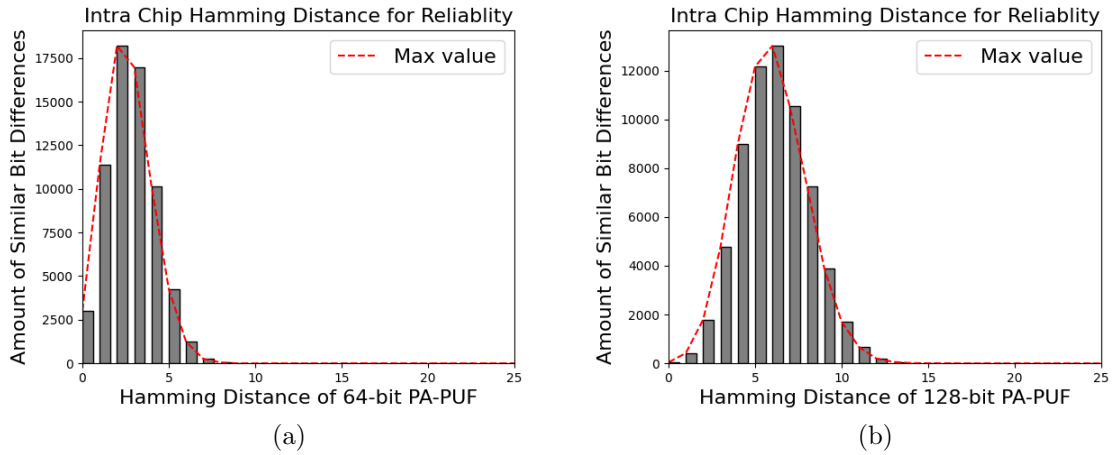


Figure 5-8: Reliability of the PA-PUF (a) 64-bit Response (b)128-bit Response.

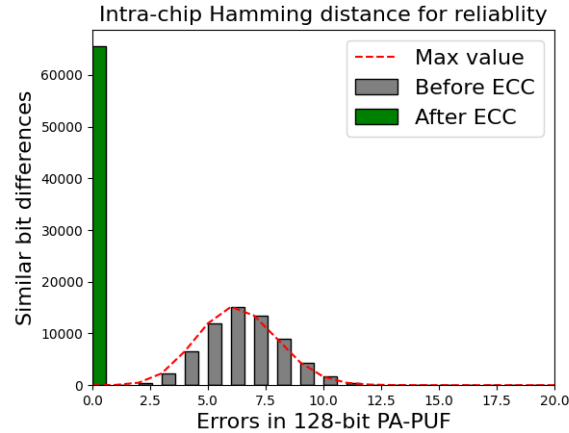


Figure 5-9: Reliability of the PA-PUF using BCH error correction codes; Resulting reliability of 100%.

ally, for the same challenge (C) different boards should give two distinct responses R_1 , and R_2 . Ideally, the inter-chip Hamming distance between R_1 , and R_2 should be 50%. The Hamming distance between two responses of two boards is known as the inter-chip Hamming distance. Figure 5-10 presents the inter-chip Hamming distance between of the PA-PUF. Moreover, the peak of the inter-chip Hamming distance of 64-bit response as shown in Figure 5-10a is at 31 while for the 128-bit response in Figure 5-10b is at 62. This shows the 128-bit PA-PUF has a uniqueness of 49.6%, whereas the ideal value of uniqueness is 50%. The next metric of interest is bit-aliasing. Since the responses were generated from hardware, the influence of the power supply (logic '1') and ground (logic '0') have to be estimated in the response.

5.5.5 Bit-Aliasing

Bit-aliasing can be defined as the influence of the power supply or ground on the response. The response should be derived from the given challenge using the variations in the circuit. In some cases, the response bit might be connected permanently to logic '1' or logic '0', which will not have any relation neither with the circuit nor with the given challenge. To estimate the bit-aliasing, the number of 0's and 1's in a particular bit were calculated. Ideally, every bit should generate an equal number of 0's and 1's. In particular, the average bit-aliasing of 128-bit PA-PUF is 49.6%, whereas the ideal value is 50%.

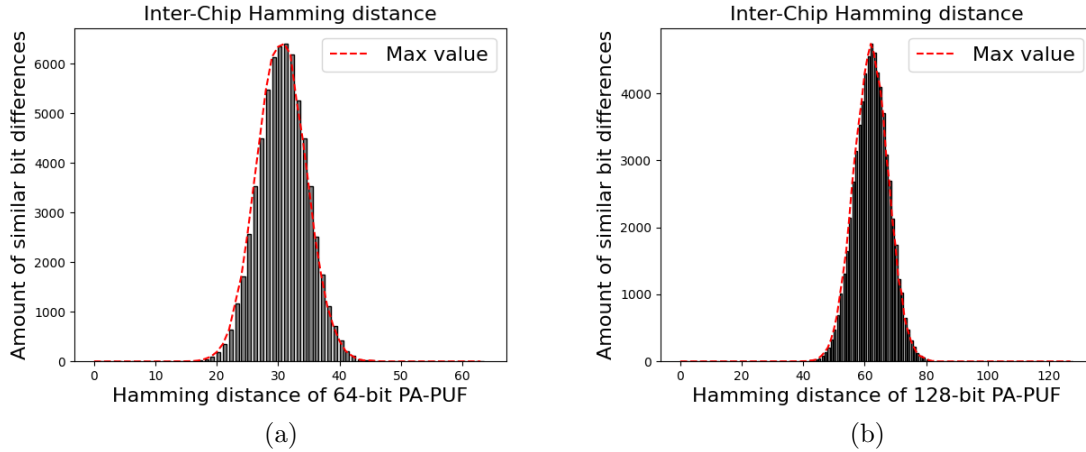


Figure 5-10: Inter-Chip Hamming Distance of the PA-PUF (a) 64-bit Response (b)128-bit Response.

Parameter %	PA-PUF	Original	CHAR	CHAR& MAJ	APUF	MPUF	CA-PUF
		Design [GHO17]					
Uniqueness	49.63	48.52	45.60	45.60	50.01	50	49.7
Uniformity	49.45	51.06	50.60	50.54	50.42	49.95	50.84
Reliability	100	92.00	98.87	99.58	99.9	99.70	99.9

Table 5.6: Comparison of the performance metrics of the ring oscillator PUF;

Ideal value of reliability is 100% while the remaining parameters have an ideal value of 50%.

5.5.6 Comparisons

Table 5.6 presents the performance metrics of the PA-PUF in terms of uniqueness, uniformity, and reliability. The PA-PUF has a comparable performance over the existing designs. Whereas the hardware resources are compared in Table 5.7 along with uniqueness and reliability. Since the number of LUTs and the number of sources available on FPGA vary over the technology, the FPGA used for the results is also reported in Table 5.7.

5.5.7 Machine Learning-based Modeling Attacks

The delay-based PUF can be modeled mathematically using machine-learning attacks on a set of challenge-response pairs as reported in [RSS⁺13b], [SSM⁺14], [EHFCS21], [SBC19a]. With a given set of known challenge-response pairs, the model should predict the response for the unknown challenge with a reasonable prediction accuracy. Over the years, various authors have shown that mathematical modeling of a classical

PUF Design ^{response size}	U (%)	R (%)	Hardware	Area
PA-PUF ¹²⁸	>49.63	100	Artix-7	47 slices per PUF
SRAM PUF ¹²⁸ [GKST07]	49.97	88	FPGA	4800 SRAM bits
Latch PUF ¹²⁸ [YSI+11]	46	87	Spartan-3	2x128 slices
Flip Flop PUF ⁴⁰⁹⁶ [MTV08]	50	95	Virtex-2	4096 flip flops
Butterfly PUF ⁶⁴ [KGM+08]	50	94	Virtex-5	130 slices
Ring Oscillator PUF ¹²⁸ [SD07]	46.15	99.52	Virtex-4	1024 ring oscillator
CRO PUF ¹²⁷ [MSE10]	43.50	96	Spartan-3	64 slices
PUF ID ¹²⁸ [GMO14]	49.90	93.93	Artix-7	128 slices per PUF
Ultra-Compact PUF ID ¹²⁸ [GO15]	49.93	93.96	Spartan-6	40 slices
Improved PUF-ID ¹²⁸ [GHO17]	45.60	99.42	Spartan-6	128 slices
APUF ⁶⁴ [HYKS10]	36.75	98.28	Virtex5	129 slices
Improved APUF ⁶⁴ [GLC+21]	19.46	97.03	Artix7	128 slices
FF-APUF ⁶⁴ [GLC+21]	41.53	97.10	Artix7	128 slices

Table 5.7: Comparisons of different PUF designs; ‘U’ is the uniqueness and ‘R’ is the reliability.

Arbiter-based PUF is easy and reports higher prediction accuracy. The challenge-response pairs of the PA-PUF have been analyzed mathematically, resulting in a prediction accuracy of 56%. Table 5.8 presents the comparison of the prediction accuracy of various Arbiter-based PUF.

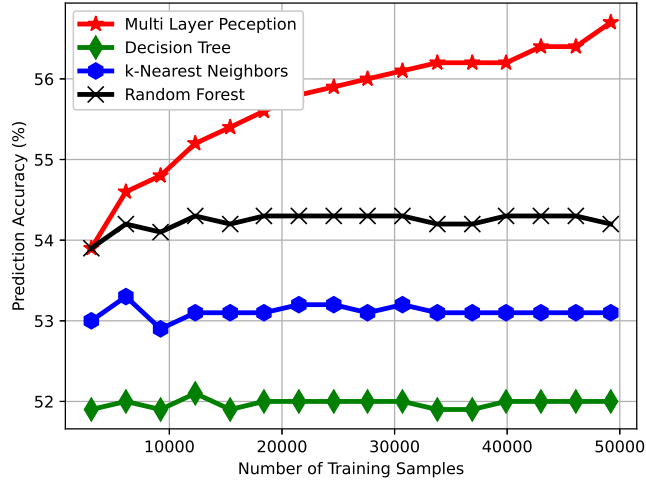


Figure 5-11: Learning Accuracy for Various Machine Learning Attacks on the PA-PUF

5.6 Conclusion

In this chapter, we delve into the analysis of PA-PUF. Initially, we construct a mathematical model for PA-PUF. We then investigate effective methods for discerning

Type of PUF	CRP	Prediction rate
Standard Arbiter PUF [EHFCS21]	5570	99%
XOR Arbiter [EHFCS21]	500,000	99%
Lightweight secure PUF [EHFCS21]	10000000	99%
Feed-forward Arbiter [EHFCS21]	50000	97%
1-XOR PUF [MSM+24]	160000	99%
MPUF [MSM+24]	150000	99%
CAPUF [MSM+24]	180000	98%
PA-PUF	52000	56%

Table 5.8: **Prediction Accuracy of Various Arbiter-based PUF.** (lower prediction accuracy represents the PUF has more robustness against modeling attacks.)

whether a given Boolean function truth table can be replicated by a PA-PUF, introducing an algorithm for this purpose. Additionally, we examine the bias in the output bit of PA-PUF when specific challenge input bits are inverted. In the final part, we have performed a detailed analysis on cryptographic properties of PA-PUF and we have observed that PA-PUF has better cryptographic properties over the other existing PUFs. The results show that the PA-PUF has more robustness against modeling attacks.

Revisiting *BoolTest* – On Randomness testing using Boolean functions

Contents

6.1	Introduction	124
6.2	Critical evaluations of Z-score	124
6.3	Finding the best Boolean function to have maximum Z-score	127
6.4	Results	136
6.5	Conclusion	139

Pseudo-random number generation is crucial in cryptology and other areas related to information technology. In a broad sense, the security of a protocol relies on the ‘randomness’ provided by the pseudo-random number generators. It is thus important to examine whether a random-looking stream has some kind of non-randomness in it. Here we consider that a binary stream is divided into blocks of a certain length m and we try to identify an m -bit Boolean function in this regard that is optimal to provide the highest Z -score for the output stream generated by the said function. In this regard, we show certain limitations of the *BoolTest* strategy by Sýs et al (2017) and present combinatorial results related to identifying the most suitable Boolean functions. We show that the existing works related to *BoolTest* identify the Boolean functions that are sub-optimal, constrained by the low degree in the Algebraic Normal Form. Our results find out the best Boolean function in this regard that will produce the maximum Z -score and the complexity is $O(N \log N)$ on the amount of random-

looking stream of length N that we read during the evaluation process. We present substantial experimental evidence corresponding to our theoretical ideas. While we solve certain combinatorial problems related to *BoolTest*, the caveat is, this test is not sufficient to conclude on randomness or non-randomness of a given stream of data.

6.1 Introduction

Random number generators find extensive utility in communication and cryptography. However, classical computers operate deterministically, making it impossible to generate true randomness beyond the influence of an initial random seed, if present. The primary innovation lies in the development of Pseudo Random Number Generators (PRNGs), where a small seed (possibly sourced from a random origin) is used as input, after which a deterministic algorithm generates a sequence of data that appears random. This generated data is not inherently random; rather, its randomness hinges solely on the initial seed.

In this regard, we concentrate on the *BoolTest* [SKŠ17], where each block of the bit-stream is applied to a suitably chosen Boolean function and the output bits are studied. In a later work, by Sýs et al [SKKŠ19], a similar technique has been used and many experimental results have been provided. As mentioned in [SKKŠ19], the *BoolTest* is a generalization of the frequency mono-bit test [BRS⁺10]. In this chapter, we revisit the techniques presented in [SKŠ17], identify certain limitations of the test, and then provide some techniques to optimize the method.

6.2 Critical evaluations of Z -score

In this section, we analyze a few issues related to the values that we receive from Z -score.

6.2.1 Z -score for data with all and equal frequency inputs

Let us consider $N = nm$ many bits of data, where $n = s2^m$, and each m -bit pattern has frequency s in the data stream, in any order. Denote such a data stream as $D_{all,m,s}$. Then we have the following result.

Lemma 7. *The Z -score for the input stream $D_{all,m,s}$ would be 0 considering any m -bit Boolean function.*

Proof. If we parse through the bit stream and generate a frequency table of all m -bit blocks then we know we will get all the patterns from $\{0,1\}^m$ equal number of times (here s) hence our distribution of inputs is uniform. Z -score for a function f shows how the output bit pattern is different for a particular distribution inferred from the data, from that of a uniform distribution. In this case, the probability of getting any m -length block in $D_{all,m,s}$ is,

$$p(x) = \frac{1}{2^m} = q(x), \forall x \in \{0,1\}^m \quad (6.1)$$

where $q(x)$ is the probability of getting any input in the uniform distribution. Then the Z -score would be,

$$z_f = \frac{n}{\sqrt{nq_f(1-q_f)}} \left| \sum_{x:f(x)=1} (p(x) - q(x)) \right| = 0. \quad (6.2)$$

Thus, the Z -score is 0 independent of the choice of the function. \square

Now, consider a simple counter circuit that generates all the m -bit patterns in a cycle, in the increasing order of decimal digits 0 to $2^m - 1$, and continues from 0 again. This data should not be considered random, but the Z -score will always be zero when we have a multiple of full cycles. The result is the same when the data comes in a cycle, but according to some permutation of 0 to $2^m - 1$. However, there are many other tests, such that linear complexity analysis, that can obtain the simplest LFSRs to distinguish among such streams. This is not possible using Z -score.

6.2.2 Maximum Z -score for frequencies s and $s + 1$

Let us consider $N = nm$ many bits of data, where $n = s2^m + u$. There are u many m -bit patterns with frequency $s + 1$ and the rest are having the frequency s , in any order. Denote this data stream as $D_{all-two,m,s}$. Let f_i denote the Boolean function with the highest z -score among those functions that output 1 for exactly i inputs. As we have denoted, z_{f_i} is the Z -score of f_i . We prove later in Section 6.3 that the truth table of f_i will contain 1 in the top i most occurring m -bit patterns. We claim that

for the stream $D_{all-two,m,s}$, the maximum Z -score would be obtained for $i = u$. Here, #1 would be

$$\#1 = \begin{cases} i(s+1) & i \leq u, \\ u(s+1) + (i-u)s & i > u. \end{cases} \quad (6.3)$$

When $i \leq u$, we put 1's in the outputs of the u patterns that occur $s+1$ times, so $\#1 = i(s+1)$. Similarly, for the case where $i > u$, the top u patterns will occur $(s+1)$ times whereas rest of the $i-u$ patterns will occur s times making $\#1 = u(s+1) + (i-u)s$. Now, we calculate z_{f_i} as,

$$z_{f_i} = \left| \frac{\#1 - nq_f}{\sqrt{nq_f(1-q_f)}} \right|, \quad (6.4)$$

where $q_f = \frac{i}{2^m}$. For simplicity of notation let $M = 2^m$. Then z_{f_i} for both cases is

$$z_{f_i} = \begin{cases} \left| \frac{i(s+1) - \frac{ni}{M}}{\sqrt{n\left(\frac{i}{M}\right)\left(1-\frac{i}{M}\right)}} \right|, & i \leq u \\ \left| \frac{u(s+1) + s(i-u) - \frac{ni}{M}}{\sqrt{n\left(\frac{i}{M}\right)\left(1-\frac{i}{M}\right)}} \right|, & i > u \end{cases} \quad (6.5)$$

$$= \begin{cases} c_1 \sqrt{\frac{i}{M-i}}, & i \leq u, \\ c_2 \sqrt{\frac{M-i}{i}}, & i > u, \end{cases} \quad (6.6)$$

where,

$$c_1 = \frac{M-u}{\sqrt{n}} \quad c_2 = \frac{u}{\sqrt{n}}. \quad (6.7)$$

From the above result, we can see that z_{f_i} is a decreasing function for $i > u$ and it is an increasing function when $i \leq u$. By this, we can conclude that z_{f_i} will be maximum for $i = u$. Thus, maximum Z -score obtained by plugging in $i = u$ will be,

$$Z_{f_u} = \sqrt{\frac{u(M-u)}{n}}. \quad (6.8)$$

6.2.3 Maximum Z -score when some of the patterns arrive only, and only once

Let us consider $N = nm$ many bits of data, where $n = u < 2^m$. There are u many m -bit patterns with frequency 1 in any order and the rest are not appearing. Denote such a data stream as $D_{some,m,u}$. It can be seen that this data stream is a special case of the $D_{all-two,m,s}$ for $s = 0$. So by plugging in the values from the above equation, we obtain

$$Z = \sqrt{\frac{u(M-u)}{n}} = \sqrt{\frac{u(M-u)}{sM+u}} = \sqrt{M-u}. \quad (6.9)$$

This is important for large block sizes, as, for large blocks, it is very clear that only a few patterns will arrive, and most of them will arrive only once. For example, if we consider $m = 256$, then in a stream of 2^{38} bits, only 2^{30} blocks will be generated. This is a very small part of 2^{256} , and thus, each of the blocks that appear will appear generally only once, and the rest huge numbers will not appear at all. That is the reason this situation needs to be studied for practical purposes in the cases of larger block sizes.

To provide specific data, we consider 1MB (megabyte) data generated by AES (OFB mode, random IV) with block size $m = 64$ bits, i.e., 8 bytes. Since the number of blocks in data is much less than 2^{64} , the probability of getting all blocks distinct is high. The number of blocks for this data would be $n = \frac{2^{20}}{2^3} = 131072$ (because 1MB = 2^{20} bytes and 64 bits = 2^3 bytes). We have checked that the generated data had all the blocks distinct, so $u = 131072$. The maximum Z -score obtained by our implementation (see [Appendix](#)) is $4294967295.9999847 = \sqrt{2^{64} - 131072} = \sqrt{M-u}$, that matches the theory. It is important to highlight that the Z -score might be very high in such a scenario.

6.3 Finding the best Boolean function to have maximum Z -score

As described in Section 2.5.1, the *BoolTest* algorithm $\mathcal{B}(deg, m, t, k)$ searches through $\alpha = \binom{m}{deg}$ monomials of degree deg . Now the top t monomials (with high Z -scores) are considered for the second stage where k out of these are added (XORed). That

is, *BoolTest* searches through a very limited space of Boolean functions since all the terms in monomials are of the same degree. On top of that, it has a fixed number of terms k , which makes it at most $\binom{\alpha}{k}$ candidate functions. This limited function search space significantly fails to discover the function with the best Z -score. As described in [SKŠ17], the Z -score is considered to heuristically find the good Boolean functions.

In this section, we will present that one can devise a deterministic algorithm to discover the Boolean function that will indeed provide the highest possible Z -score. It is computationally elusive to exhaustively search all the 2^{2^m} Boolean functions for $m \geq 6$. We demonstrate an $\mathcal{O}(N \log N)$ algorithm for data size N , to achieve this for any arbitrary block size m . So it would be possible to run this algorithm on almost any size of data that can be stored (and read in reasonable time) in a particular machine.

In this regard, let us first define some notations and prove a few technical results. We define MS for a function f as follow:

$$MS(f) = |p_f - q_f|, \quad (6.10)$$

where p_f and q_f are same as defined earlier. If there are t inputs for which we get 1 as output then, $q_f = \frac{t}{2^m}$. We first show how using this metric we will obtain the function with the highest Z -score in $\mathcal{O}(m2^m)$ time (an improvement over $\mathcal{O}(2^{2^m})$). Define \mathcal{F}_t to be the set of all m -variable Boolean functions that will output 1 on exactly t of the 2^m possible inputs. Thus, for all functions $f \in \mathcal{F}_t$, we have

$$q_f = \frac{t}{2^m} \quad (6.11)$$

Since we have fixed the value q_f for a given set \mathcal{F}_t , we have essentially fixed the denominator of the Z -score. Thus it would be easy to maximize it among the functions in this set. Further, if \mathcal{F} is a set of all Boolean functions of m variables,

$$\mathcal{F} = \bigcup_{t=0}^n \mathcal{F}_t \quad (6.12)$$

Let $f_t \in \mathcal{F}_t$ be the function with maximum Z -score in the set \mathcal{F}_t . So the Boolean

function with the best Z -score in \mathcal{F} would be,

$$f = \arg \max_{f_t} z(f_t), \quad (6.13)$$

where, $z(f_t)$ gives the Z -score for the function f_t . We claim that we can find the function with maximum MS in each \mathcal{F}_t (say $f_t \in \mathcal{F}_t$) efficiently. We further claim that the same f_t will provide us with the highest Z -score inside each \mathcal{F}_t .

Let us first present a technical result. The complement of a Boolean function f will have a truth table with negated outputs for each input. In other words, if f is a Boolean function and f' is its complement then for input x

$$f'(x) = 1 \oplus f(x) \quad (6.14)$$

So if a Boolean function f of m -input, outputs 1 for t -inputs then f' will output 1 for the other $2^m - t$ inputs. Now, we prove that the Z -score for both f and f' would be the same.

Proposition 14. *The Z -score for a function f and its complement f' would be the same.*

Proof. Let us assume that f is an m -input Boolean function that will output 1 t times then,

$$z_f = \frac{n}{\sqrt{nq_f(1-q_f)}} \left| \sum_{x:f(x)=1} p(x) - \frac{t}{2^m} \right| \quad (6.15)$$

where $p(x)$ is the proportion of input x in the input data. Now, $\sum_{x:f(x)=1} p(x) = p_f$. This can also be written as,

$$z_f = \frac{n}{\sqrt{nq_f(1-q_f)}} \left| \left(1 - \sum_{x:f(x)=0} p(x) \right) - \frac{t}{2^m} \right| \quad (6.16)$$

By definition, if f outputs 0 for some input x , then for the similar inputs f' outputs 1, then

$$z_f = \frac{n}{\sqrt{nq_f(1-q_f)}} \left| \left(1 - \frac{t}{2^m} \right) - \sum_{x:f'(x)=1} p(x) \right| \quad (6.17)$$

This is the Z -score for f' as,

$$z_f = \frac{n}{\sqrt{nq_f(1-q_f)}} \left| \sum_{x:f'(x)=1} p(x) - \left(\frac{2^m-t}{2^m}\right) \right| \quad (6.18)$$

$$= \frac{n}{\sqrt{n(1-q_{f'})q_{f'}}} \left| \sum_{x:f'(x)=1} p(x) - \left(\frac{2^m-t}{2^m}\right) \right| = z_{f'} \quad (6.19)$$

□

Now we present the main result.

Theorem 13. *If f_t is a function in \mathcal{F}_t and $MS(f_t) = \max_{f \in \mathcal{F}_t} MS(f)$ then*

$$f_t(x) = \begin{cases} 1 & \text{if } x \in A_t \\ 0 & \text{otherwise} \end{cases} \quad (6.20)$$

where A_t is the set of t -most occurring inputs in the data file.

Proof. For any $f \in \mathcal{F}_t$ we have,

$$MS(f) = |p_f - q_f|$$

Our goal is to maximize MS and find the function f for which we get the maximum score. As the function f outputs 1 on exactly t inputs, we have

$$MS(f) = \left| p_f - \frac{t}{2^m} \right| \quad (6.21)$$

Since we have fixed t , maximizing $MS(f)$ means either maximizing p_f or minimizing p_f .

Following Proposition 14, we show that maximization is enough, i.e., it is not necessary to minimize the sum p_f and this can be seen by the fact that a function and its complement share the same Z -score. Let's say we minimize the value of p_f and f gives 1 for t inputs. Since we are minimizing the summation what we are doing is taking the least frequent t inputs and making the function f output 1 on these inputs. In other words, f outputs 0 for the top $2^m - t$ inputs, so f' (the complement of f) outputs 1 for those $2^m - t$ inputs. Since $f' \in \mathcal{F}_{2^m-t}$, we can calculate the

function $g \in \mathcal{F}_{2^m-t}$ that has the best Z -score in \mathcal{F}_{2^m-t} . If $g = f'$, then we do not need to minimize p_f because we would anyway find the function f' , and calculating the Z -score for f would not be needed. If $g \neq f'$, then also we do not need to minimize p_f , because there is already a function g , that has Z -score greater than that of f . Thus we do not need to minimize the summation separately as we gain information about a function's complement from the function itself.

For a given input file, we calculate the number of occurrences (or probability of occurrence) of each m -bit block [0 to $2^m - 1$]. We will sort an array containing the 2^m m -bit blocks (consider truth table rows) by their proportion in non-decreasing order. For each t , we construct a function f_t that returns 1 on the top t highest probability blocks and 0 for the rest.

For any other function f'_t that outputs 1 on exactly t inputs, $p_{f'_t} \leq p_{f_t}$ because we are summing over the highest $p(x)$. Note that there might be other functions f'_t with $p_{f'_t} = p_{f_t}$, but $p_{f'_t}$ can never be greater than p_{f_t} . \square

Now that we have established a way to find the maximum MS , we use it to calculate the Z -score. The relation between MS and Z -score can be shown as:

$$\begin{aligned} z_f &= \left| \frac{\#1 - nq_f}{\sqrt{nq_f(1-q_f)}} \right| = \left| \frac{np_f - nq_f}{\sqrt{nq_f(1-q_f)}} \right| = \frac{n}{\sqrt{nq_f(1-q_f)}} |p_f - q_f| \\ &= \frac{n}{\sqrt{nq_f(1-q_f)}} MS(f). \end{aligned} \tag{6.22}$$

Fixing the set F_t from which the function will be chosen, we know that the term $\frac{n}{\sqrt{nq_f(1-q_f)}}$ is constant and so if a function maximizes MS , it maximizes the Z -score. So, we find the highest Z -score for each \mathcal{F}_t , $t = \{1, \dots, 2^m - 1\}$. Note that, for $t = 0$ or $t = 2^m$ only constant functions are possible with undefined Z -scores, that we will not consider. Then we find the maximum among these $2^m - 1$ Z -scores. Based on this, we have the following algorithm.

In Algorithm 5, to get the actual Boolean function providing the highest Z -score, we can sort the inputs to the truth table using their probabilities and output the

truth table with 1's in the t_{max} highest probability inputs.

Algorithm 5: modified boottest (run time: $\mathcal{O}(2^m)$)

```

/* Suppose  $\mathcal{S}$  is the set of inputs for which the function under
   consideration returns 1 (this set completely defines the function) */
1
2  $I \leftarrow 0$  to  $2^m - 1$  (truth table inputs);
3  $P \leftarrow$  probability of occurrence of each  $m$ -bit block according to data;
4  $z \leftarrow 0$  // The  $Z$ -score value
5  $t \leftarrow 1$  // Number of ones as output
6  $p \leftarrow 0$ 
7 SORT ( $I, P$ ) // Sort array of truth-table-inputs wrt their probability of
   occurrence
8 while  $t < 2^m$  do
   | // try to maximize  $|p - q|$ 
9   |  $p \leftarrow p + P[t - 1]$  // Adding next most occurring input to  $\mathcal{S}$ 
10  |  $q \leftarrow \frac{t}{2^m}$ 
11  |  $z_t \leftarrow n \cdot (p - q)$ 
12  |  $z_t \leftarrow \frac{z_t}{\sqrt{nq(1-q)}}$ 
13  | if  $z_t > z$  then
14  | |  $t_{max} \leftarrow t$ 
15  | |  $z \leftarrow z_t$ 
16  |  $t \leftarrow t + 1$ 
17 end

```

6.3.1 Improving the time and space complexity further

The main drawbacks of the above algorithm are as follows:

- It takes $\mathcal{O}(m2^m + N)$ time, $\mathcal{O}(N)$ for calculating the probabilities, $\mathcal{O}(m2^m)$ for sorting the inputs according to the probabilities, and $\mathcal{O}(2^m)$ iterations in the loop.
- It would also take $\mathcal{O}(2^m)$ space for storing the probability array.

Note that N is the length of the data in bits, and thus, we have to accept that for analysis. On the other hand, if m is large, such as $m = 128$, then the above algorithm cannot be executed with the present computational power. Thus, we now improve the algorithm so that it requires $\mathcal{O}(N \log N)$ time and $\mathcal{O}(N)$ space.

Some observations.

Suppose all the 2^m possible blocks do not appear in the input data. This is natural in the case where the block size is large. If we have data of the order of 2^{40} (say 2^{32} blocks of length 2^8 each), then the block size being 256 bits, it is very clear that at most 2^{32} different patterns may appear. Thus, the algorithm should be redesigned. We already considered that there are n blocks in the data, i.e., $n = \frac{N}{m}$. Now let us consider there are d distinct m bit patterns, i.e., $d \leq n$. That is, there are d data blocks that have non-zero probabilities of occurrences. Now we explain that as in the algorithm above, we should not check all t . Rather, increasing t above d is not required as the Z -score for those values of t would not be more than the Z -score achieved for $t = d$.

Theorem 14. *Let, d be the number of distinct blocks that appear in the data and z_t be the highest possible Z -score for a function $f_t \in \mathcal{F}_t$. Then, $z_{f_d} > z_{f_j}, \forall j > d$.*

Proof. For block size m and fixed input data having n many blocks (may not be all distinct), let, f_t be the function with highest Z -score in \mathcal{F}_t and z_{f_t} be the Z -score corresponding to f_t . For $t = d$, consider the truth table of the function f_d providing the highest Z -score z_{f_d} . By Algorithm 5, we know that there would be 1's in the d highest probability blocks in the truth table, which would be all the blocks with non-zero probability. Thus, for every possible data block that appears in the data, there would be a 1 in the corresponding row in f_d 's truth table. So, #1, i.e., the number of data blocks in the input which when fed into the function f_d would return 1 is n , the number of blocks. Thus the Z -score will be:

$$z_{f_d} = \left| \frac{n - \frac{d}{2^m}n}{\sqrt{n \frac{d}{2^m} (1 - \frac{d}{2^m})}} \right| = \sqrt{n} \left| \frac{1 - \frac{d}{2^m}}{\sqrt{\frac{d}{2^m} (1 - \frac{d}{2^m})}} \right| \quad (6.23)$$

When we increase t , by the above algorithm, the truth table corresponding to each f_t would contain 1's in t highest probability blocks. Hence, it would contain 1's in all

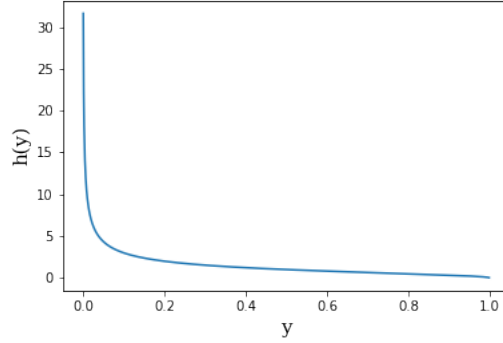
the blocks with non-zero probability (since there are $d < t$ such blocks), i.e, all blocks that appear in the data will have 1 in the corresponding truth-table row. Thus, #1 will remain fixed at n . Hence the Z -score will be:

$$z_{t>d} = \left| \frac{n - \frac{t}{2^m}n}{\sqrt{n\frac{t}{2^m}(1 - \frac{t}{2^m})}} \right| = \sqrt{n} \left| \frac{1 - \frac{t}{2^m}}{\sqrt{\frac{t}{2^m}(1 - \frac{t}{2^m})}} \right| \quad (6.24)$$

Now, the function,

$$h(y) = \left| \frac{1 - y}{\sqrt{y(1 - y)}} \right| \quad (6.25)$$

is a decreasing function for $y \in (0, 1)$. The graph of h is shown below:



So, for $t > d$, we have: $\sqrt{nh}(\frac{d}{2^m}) > \sqrt{nh}(\frac{t}{2^m})$, i.e.,

$$\sqrt{n} \left| \frac{1 - \frac{d}{2^m}n}{\sqrt{\frac{d}{2^m}(1 - \frac{d}{2^m})}} \right| > \sqrt{n} \left| \frac{1 - \frac{t}{2^m}n}{\sqrt{\frac{t}{2^m}(1 - \frac{t}{2^m})}} \right|,$$

i.e., $z_{fd} > z_{ft}$, for $t > d$. Thus the proof. \square

Based on the above result, we obtain a more efficient algorithm. Now t will be considered till d which is $\mathcal{O}(N)$ and not up to 2^m like in the previous method. Since we are considering large m now, we cannot use the earlier idea of having 2^m length array for storing the probability of each m -bit block or storing and sorting all the truth table rows. Instead, we sort the m -bit blocks in the input data interpreting their natural decimal values. This brings the same blocks together. Then we can count the number of occurrences of each block by counting the number of consecutive similar blocks, and the number of distinct blocks that are appearing in the data.

Function Encoding.

Algorithm 6: optimized modified boottest (run time: $\mathcal{O}(N \log N)$)

```
// Sort data by value of m bit blocks to bring same blocks together
1 SORT(Data)
2  $m \leftarrow$  block size
3  $n \leftarrow$  number of blocks in data
4 count  $\leftarrow$  1
5 idx  $\leftarrow$  0
6 DistinctBlocks  $\leftarrow$  empty  $N$  length array of  $m$ -bit blocks
7 Occurrences  $\leftarrow$   $N$  length array of integers initialized to 0s
8 for  $i \leftarrow 1$  to  $N$  do
9   if  $i = N$  or  $Data[i] \neq Data[i-1]$  then
10     Occurrences[idx]  $\leftarrow$  count
11     DistinctBlocks[idx]  $\leftarrow$  Data[ $i - 1$ ]
12     count  $\leftarrow$  0
13     idx  $\leftarrow$  idx + 1
14   count  $\leftarrow$  count + 1
15 end
16  $d \leftarrow$  idx //  $d$  is the number of distinct blocks in data
17 SORT-BY-OCCURRENCES(DistinctBlocks, Occurrences)
18  $\Delta \leftarrow 0$ 
   //  $\Delta$  is the number of occurrences of the blocks in Data, for which the
   // candidate function being considered returns 1
19  $z_{max} \leftarrow -1$ ;  $t_{max} \leftarrow -1$ 
20 for  $i \leftarrow 0$  to  $d - 1$  do
21    $t \leftarrow i + 1$ 
22    $\Delta \leftarrow \Delta + \text{Occurrences}[i]$ 
23    $p \leftarrow \Delta / N$ 
24    $q \leftarrow t / 2^m$ 
25    $z \leftarrow \left\lfloor \frac{N(p-q)}{\sqrt{Nq(1-q)}} \right\rfloor$ 
26   if  $z > z_{max}$  then
27      $z_{max} \leftarrow z$ 
28      $t_{max} \leftarrow i + 1$ 
29 end
30 return  $z_{max}$ , list of top  $t_{max}$  inputs of DistinctBlocks
```

Note that representing an arbitrary m -input Boolean function would require $O(2^m)$ space in the worst case, irrespective of the function encoding scheme such as truth table or ANF, etc. However, in our case, the algorithm ensures that whatever is the highest Z -score Boolean function, it would only output 1 for at most d inputs. So, consider the encoding scheme where we just list the inputs (truth table rows) for which the function returns 1, that is we will work with the support set of the function. This will require $O(N)$ space. Based on all these we present an efficient algorithm.

6.4 Results

In this section, we present the experimental results and compare them with the existing works.

6.4.1 RC4

It is well known [MS02] that the second byte of the RC4 keystream is biased towards zero with probability almost $\frac{2}{256}$, which is significantly higher than the uniform random value $\frac{1}{256}$. Consider that a long keystream byte sequence is generated with randomly chosen secret keys and then accumulates the second output bytes of RC4 in each case. Since the probability of any other value except zero is slightly less than $\frac{1}{256}$, according to our strategy, the Boolean function that should provide the highest Z -score should have output 1 for all zero input and the rest of the outputs should be 0. This function contains all the terms in ANF.

Based on the bias, it can be shown that the best distinguisher Boolean function for this data, working on 8-bit blocks is the one that returns 1 on the $(0, 0, 0, 0, 0, 0, 0, 0)$ input and 0 on everything else (say f_0). The complement of this function may also be considered. Let us name the input variables as (x_0, x_1, \dots, x_7) for the eight-bit block. The ANF of the function to maximize the Z -score contains all the terms in ANF, as provided by our Algorithm 6 in all the runs with different sets of data. It is clear that the ANF is quite complicated and such an ANF will never be considered for BoolTest [SKŠ17]. We note that taking the constraint of degree 3, *BoolTest* [SKŠ17] provides different functions in different runs towards the sub-optimal efforts in maximizing the Z -score, i.e., cannot provide the correct answer due to sub-optimality. In Table 6.1, B1 is BoolTest [SKŠ17] with parameters (degree = 2, combine-degree = 2) and B2

File	B1		B2			Bool-Test-2	
	highest Z-score	best-distinguisher	highest Z-score	best-distinguisher	highest Z-score	best-distinguisher	
RC4 1MB	4.55	$x_3x_6 + x_3x_4$	4.47	$x_1x_3x_5$ + $x_3x_6x_7$ + $x_3x_4x_7$	64.16	f_0	
RC4 10MB	11.56	$x_2x_6 + x_3x_7$	9.90	$x_3x_4x_7$ + $x_0x_2x_7$ + $x_2x_4x_6$	204.36	f_0	
RC4 100MB	31.05	$x_6x_7 + x_4x_5$	23.64	$x_2x_6x_7$ + $x_3x_4x_5$ + $x_1x_4x_6$	643.14	f_0	

Table 6.1: Testing RC4 2nd byte samples.

represents BoolTest [SKŠ17] run with parameters (degree = 3, combine-degree = 3). Our results are presented with Bool-Test-2, where it could be seen that the Z -score is much higher.

6.4.2 Comparison with Java rand and AES [DR02, AES01]

In this section, we show a comparison of the Z -scores that we have obtained from Java Random and AES. We use 10MB files for both AES and Java Random and apply *BoolTest* [SKŠ17] as well as our Algorithm 6.

File	Block-size	B_1	B_2	Best Z -score
Java 10MB	8	2.668	3.8144	11.7884
Java 10MB	32	4.2611	5.8122	65526.278
Java 10MB	256	5.6797	9.3637	3.4×10^{38}
AES 10MB	8	3.4817	4.7543	12.1083
AES 10MB	32	4.322	6.289	65526.10
AES 10MB	256	5.557	8.4509	3.4×10^{38}

Table 6.2: Results

In the Table 6.2 above, B_1 is the *BoolTest* algorithm with parameters $\mathcal{B}(deg = 2, m = \text{Block-size}, t = 128, k = 2)$ and B_2 is the *BoolTest* algorithm with parameters $\mathcal{B}(deg = 3, m = \text{Block-size}, t = 128, k = 3)$. As explained in this initiative, since *BoolTest* [SKŠ17] searches in limited function space, the Z -score obtained with those

constraints is sub-optimal. On the other hand, we obtain very high scores in this regard. Our analysis in Section 6.2.3 theoretically explains why such large values in Z -score are possible.

6.4.3 Cross-testing by the generated polynomials, i.e., functions

The motivation is to generate the Boolean function for which the Z -score will be maximized so that one can interpret a high value outside some interval as non-randomness. One interesting methodology to evaluate this *BoolTest* is to generate the best function from one data set and to use that function to evaluate the Z -score of another random-looking data set. First, let us consider the *BoolTest* [SKŠ17] heuristics in this regard.

Function generated by	Z -score				
	Java 1MB	Java 10MB	Java 100MB	AES 1MB	AES 10MB
Java 1MB ($m=8$)	3.22	0.8	1.0	0.156	1.124
Java 1MB ($m=256$)	6.08	0.22	0.80	0.148	1.977
Java 100MB ($m=256$)	2.6	8.5	36.75	1.1	2.0
AES 10MB ($m=256$)	0.56	0.30	1.34	0.02	8.45

Table 6.3: Cross-testing with *BoolTest* [SKŠ17].

In the Table 6.3, for Java 1MB ($m = 8$), we have used the parameters $\mathcal{B}(deg = 2, m = 8, t = 128, k = 2)$. For Java 1MB ($m = 256$), we have used the parameters $\mathcal{B}(deg = 2, m = 256, t = 128, k = 2)$ and for the Java 100MB ($m = 256$) we have used the parameters $\mathcal{B}(deg = 3, m = 256, t = 128, k = 3)$.

Using our Algorithm 6 (with implementation in Appendix) we performed cross-testing too and obtained the following results. The small values that we obtained for $m = 256$ are probably because most of the 256-bit blocks for which the generated function outputs 1 will never arrive in other samples of the data. The expected number of 1s will also be small.

To summarize, we generate the Boolean function for a distinguisher based on a

Function generated by	Z -score				
	Java 1MB	Java 10MB	Java 100MB	AES 1MB	AES 10MB
Java 1MB ($m=8$)	13	0.26	0.87	0.12	0.38
Java 1MB ($m=8$)	0.58	11.8	0.50	0.20	1.0
Java 100MB($m=256$)	10^{-33}	3×10^{-33}	3.4×10^{38}	9.6×10^{-34}	3×10^{-33}
AES 10MB($m=256$)	3×10^{-34}	9.6×10^{-34}	3×10^{-33}	3×10^{-34}	3.4×10^{-38}

Table 6.4: Cross-testing for our algorithm

particular sample of data. Then with this function, we run the distinguisher for a different set of random-looking data and observe the Z -score. Generally, the lower values related to AES provide the understanding that it demonstrates more randomness than the Java random number generator. This is a natural conclusion, but these kinds of cross-testing require further investigation and more concrete theoretical support.

6.5 Conclusion

In this chapter, we have studied certain limitations of *BoolTest* [SKŠ17]. We introduce combinatorial findings associated with pinpointing the optimal Boolean functions for maximizing the Z -score that was not achievable using the heuristic outlined in *BoolTest* [SKŠ17]. Our Algorithm 6 efficiently determines the optimal Boolean function with the highest Z -score in $O(N \log N)$ time, where N represents the volume of available data. While addressing specific combinatorial challenges associated with *BoolTest*, it's important to note its insufficiency in determining data stream randomness. Although statistical interpretations in [SKŠ17, Section 5] are available, we advocate for further evaluation of this tool.

Appendix : Implementation Details

For large block sizes, the Z -score would be very large and it would not be possible to store the results accurately in 64 bits data elements of C programming compilers. For example, the highest Z -score for a block size of 256 might be of the order of 10^{38} . It would require ~ 126 bits to represent such integers up to 10^{38} . To maintain accuracy,

we instead use the GNU multi-precision library (GMP) for the calculations [gmp].

Listing 6.1: C code for final algorithm

```
// data = address (of first byte) of nm bit data
// len = length in bytes of data, i.e., nm/8
// m = block size
void generate(unsigned char* data, int len, int m)
{
    unsigned long n = (len*8)/m;

    // merge sort m-bit blocks at address 'data'
    // by the value of m-bit blocks
    sort_large_block(data, 0, n - 1, m);

    if(n == 0)
    {
        printf("no_data\n\n");
        return;
    }

    // need length(occurences) = num_distinct_blocks (<= n)
    int* occurences = malloc(n * sizeof(int));
    // need length(distinct_blocks) = num_distinct_blocks * size_of_block (<= data size)
    unsigned char* distinct_blocks = malloc(len * sizeof(unsigned char));

    unsigned long num_distinct_blocks = 0;
    unsigned long curr_count = 1, idx = 0;
    unsigned long sum_occurences = 0;
    for(unsigned long i=1; i<=n; i++)
    {
        if(i==n || !same_block(data, i, data, i-1))
        {
            copy_block(distinct_blocks, idx, data, i-1, m);
            occurences[idx] = curr_count;
            sum_occurences += occurences[idx];
            curr_count = 0;
            idx++;
        }
        curr_count++;
    }
    num_distinct_blocks = idx;

    // merge sort m-bit blocks at address 'distinct_blocks'
    // by their number of occurences in the data
    sort_by_occurences(distinct_blocks, occurences, 0, num_distinct_blocks - 1, m);

    mpf_t z_max;
    mpf_init(z_max); mpf_set_ui(z_max, (unsigned long) 0);
    mpf_t t, q, p, MS, z, d1, d2;
    mpf_init(t);
    mpf_init(q);
    mpf_init(p);
    mpf_init(MS);
    mpf_init(z);
    mpf_init(d1);
    mpf_init(d2);

    int t_max = -1; int num_one = 0;
    sum_occurences = 0;

    for(unsigned long i=0; i<num_distinct_blocks; i++)
    {
        sum_occurences += occurences[i];

        mpf_set_ui(t, (unsigned long)(i + (unsigned long)1));
        mpf_set_ui(d1, 2); mpf_pow_ui(d1, d1, m);
```

```

if(mpf_cmp(t, d1) == 0)
{
printf("t = 2^m: break\n\n");
break;
}

// MS
mpf_div(q, t, d1);
mpf_set_ui(p, sum_occurrences);
mpf_div_ui(p, p, (unsigned long)n);
mpf_sub(MS, p, q);
mpf_abs(MS, MS);

// z-score
mpf_mul_ui(MS, MS, (unsigned long) n);
mpf_set_ui(d2, (unsigned long) 1);
mpf_sub(d2, d2, q);
mpf_mul(d2, d2, q);
mpf_mul_ui(d2, d2, (unsigned long) n);
mpf_sqrt(d2, d2);
mpf_div(z, MS, d2);

if(mpf_cmp(z, z_max) > 0)
{
mpf_set(z_max, z);
t_max = i+1;
num_one = sum_occurrences;
}

}

// get anf
if(m <= 16)
{
int x = m - 3; if(x < 0) x = 0;
unsigned char* truth_table = (unsigned char*)malloc((1<<x)*sizeof(unsigned char));

for(int i=0; i<(1<<x); i++)
truth_table[i] = 0;

for(int i=0; i<t_max; i++)
set_bit(truth_table, get_block_as_int(distinct_blocks, i));
anf_from_truth_table(truth_table, m);
free(truth_table);
}

printf("highest z-score ---\n");
mpf_out_str(stdout, 10, 0, z_max);
printf("\n t_max=%d, num_one=%d\n\n", t_max, num_one);

save_bool_function(distinct_blocks, num_distinct_blocks, t_max, m);
free(occurrences);
free(distinct_blocks);
}

```


Analysis of Boolean Functions Related to Two-party Nonlocal Games

Contents

7.1 Introduction	144
7.2 Inconsistency for the $2 + 2$ Partition and Its Subpartitions	145
7.3 Results related to classical strategies for any n -party nonlocal game	147
7.4 Some Basic Results	149
7.5 Analysis of the Maximum Success Probability in Classi- cal Scenario	159
7.6 Analysis of the Maximum Success Probability in Quan- tum Scenario	160
7.7 Analysis of the Results in [DPM19]	163
7.8 Analysis of the Binary Input Binary Output Two-party Nonlocal Games	165
7.9 Conclusion	183

In this chapter, we thoroughly investigated all Boolean functions involving four variables to model binary input binary-output two-party nonlocal games, assessing their performance in both classical and quantum settings. Our analysis identifies several games, apart from the CHSH game, that exhibit greater success probabilities in quantum scenarios than in classical ones. This underscores the effectiveness of the CHSH game and similar partitioned games in distinguishing between quantum

and classical methodologies. Additionally, we extend the characterization of classical strategies to any n -party nonlocal game. Though this extension does not focus on identifying quantum advantage in n -party nonlocal games, it establishes the groundwork for future analyses on classical-quantum separations.

7.1 Introduction

In a binary input-binary output two-party nonlocal game, each player has two choices for the input and two choices for the output. The most well known binary input binary output two-party nonlocal game is the CHSH game [CHSH69] where a referee provides two uniformly random bits x_1 and x_2 to each of the two players. After receiving the inputs, the two parties send their output bits x_3 and x_4 to the referee. The function that represents the CHSH game is of the form $f(x_1, x_2, x_3, x_4) = (x_1 \wedge x_2) \oplus (x_3 \oplus x_4)$. From the winning condition of the CHSH game, one can easily check that the two parties can win the game whenever the values of x_1, x_2, x_3, x_4 satisfy $f(x_1, x_2, x_3, x_4) = 0$. It is well known that the maximum success probability of the CHSH game in the classical scenario is 0.75 whereas the maximum success probability using quantum resources is $\cos^2 \frac{\pi}{8}$ (which is approximately 0.85).

There are several known two-party nonlocal games that offer quantum advantages [BBT05, CM14]. To the best of our knowledge, from the class of all possible binary input binary output two-party nonlocal games, the only known game that offers quantum advantage is the CHSH game. As the CHSH game can't be won with certainty in the quantum scenario, it would be interesting to check whether there exists any other binary input binary output two-party nonlocal game for which quantum advantage can be achieved and the game can be won with a better quantum success probability than the CHSH game (because from the analysis of [BM18], it is clear that if there exists any such game then it can be used for DI testing instead of the CHSH game to reduce the overall sample size).

In this chapter, we have explored the performance of all possible binary input binary output two-party nonlocal games (having atleast one successful outcome for each possible input) by considering them as four variable Boolean functions. We also extend the technical results related to classical strategies for any n -party nonlocal game and provide mathematically rigorous proofs. Although our findings do not provide

explicit insights into games where classical-quantum separation can be leveraged, in a forthcoming investigation targeting the characterization of n -party nonlocal games with quantum advantages, our results will establish the groundwork for identifying these specific games and developing their optimal strategies.

7.2 Inconsistency for the $2 + 2$ Partition and Its Subpartitions

It is well known that for a binary input binary output two-party nonlocal game, there are 4 possible inputs and for each input, there can have atmost 4 different outputs. It is also clear that for a particular input string (i.e., for a particular value of xy), the two players can have atmost 16 different strategies to generate their outcomes in the classical scenario. Based on the outcomes, here we classify the 16 different strategies into four groups where each group has 4 different strategies and each of these strategies leads to a different outcome for a particular input. These four groups are as follows.

Group 1 (Constant Strategies): $00, 01, 10, 11$

Group 2 (input-dependent Strategies): $xy, \bar{x}y, x\bar{y}, \bar{x}\bar{y}$

Group 3 (Mixed Strategies): $x0, x1, \bar{x}0, \bar{x}1$

Group 4 (Mixed Strategies): $0y, 1y, 0\bar{y}, 1\bar{y}$

Whenever two different inputs are chosen, there are two possibilities for their values. Either the inputs are complement to each other (i.e., of the form $xy, \bar{x}\bar{y}$) or they are not complement to each other (i.e., of the form $xy, \bar{x}y$ or $xy, x\bar{y}$).

Now if a strategy is applied to these chosen inputs, the generated output pair may match in all two positions or only in one position or none of the positions. One can easily explore that for a complement input pair, if the outputs are same then the corresponding strategy must be constant. Similarly, if the outputs are complement to each other (i.e., of the form $ab, \bar{a}\bar{b}$) for a complement input pair, the corresponding strategy must be an input-dependent strategy and if the outputs have only one different bit (i.e., of the form $ab, \bar{a}b$ or $ab, a\bar{b}$) then the corresponding strategy must be a mixed strategy (either from group 3 or from group 4). In this similar way, one can also explore the strategies for the cases where the inputs are not complement to each other.

It is interesting that whenever two different inputs match in exactly one bit position (i.e., inputs of the form $xy, \bar{x}y$ or $xy, x\bar{y}$) but the output bits in that position are different for different inputs then one can't get any strategy that satisfies atleast one output for both the inputs. More formally, whenever the inputs and the corresponding outputs are of the form mentioned in Table 7.1, one can't get any strategy that satisfies atleast one output for both the inputs.

Input	Corresponding output
xy	$ab, \bar{a}\bar{b}$
$x\bar{y}$	$\bar{a}b, ab$

Table 7.1: Inconsistent Outputs

This leads us to the following result.

Theorem 15. *For the two input-output pairs of a game, if one bit of the input pair remains the same and the corresponding bit of their outputs is different, then no strategy satisfies atleast one output for both the inputs.*

Proof. : Without loss of generality, here we assume that the input pair is of the form $xy, x\bar{y}$ and the corresponding outputs are of the form ab and $\bar{a}b$ respectively (i.e., the first bit for both the inputs are same however the first bit for the two outputs are different).

As the two outputs are different, no constant strategy can satisfy both outputs for this input pair. One can also check that whenever a mixed strategy either from group 3 or from group 4 is applied to this specified input pair, the first bit of the corresponding outputs always remains the same. However for the given outputs (as specified in table 7.1), the first bits of the outputs for the two different inputs are complement to each other. This implies that no constant or mixed strategy can satisfy atleast one output for both inputs.

Similarly, one can also explore that whenever an input-dependent strategy is applied to this specified input pair, the corresponding outputs are of the form $ab, \bar{a}\bar{b}$ or $\bar{a}b, ab$. This implies that no strategy from any of the groups can satisfy atleast one output for both the inputs. Similarly, one can also argue for the other possible input-output pairs of this form. □

From this result, it is clear that if a game has two inputs of the form $xy, x\bar{y}(xy, \bar{x}y)$ and the corresponding outputs are of the form $ab, \bar{a}\bar{b}(ab, \bar{a}b)$ and $\bar{a}b, \bar{a}\bar{b}(ab, \bar{a}b)$ respectively then there exist no strategy which satisfies atleast one output for both the inputs $xy, x\bar{y}(xy, \bar{x}y)$.

7.3 Results related to classical strategies for any n -party nonlocal game

In an n -party nonlocal game, there are n players and a referee. Each player is given an input bit $x_i \in \{0, 1\}$ by the referee and based on a pre-decided strategy $s_i = \{0, 1, x_i, \bar{x}_i\}$, the player sends back an output bit $a_i \in \{0, 1\}$ to the referee. In between, the players are strictly not allowed to communicate with other players once it receives the input bit from the referee. We can express any n -party nonlocal game in the form of a $2n$ -input 1-output Boolean function: $f(x_1, \dots, x_n, a_1, \dots, a_n)$, where $\mathbf{x} = x_1, \dots, x_n$ represents the inputs from the referee, and $\mathbf{a} = a_1, \dots, a_n$ denotes the corresponding outputs. The players win against the referee if and only if $f = 0$. Depending on the number of players (n), the referee (may) restricts the input bits x_1, \dots, x_n to a subset of $\{0, 1\}^n$, similar to the GHZ game discussed earlier [See Definition 5].

Since each player can choose any of the four possible strategies, the total number of classical strategies for any n -party nonlocal game is given by 4^n . Note that the input bits received by the players can be denoted by an n -bit string $\mathbf{x} = x_1, \dots, x_n$. Similarly, the individual strategies of all n players can be denoted by $\mathbf{s} = s_1, \dots, s_n$, and the corresponding output is represented by $\mathbf{a} = a_1, \dots, a_n$. In this case, we denote input bit-strings by \mathbf{x}, \mathbf{y} , output bit-strings by $\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}$, and the corresponding strategies by $\mathbf{s}, \mathbf{t}, \mathbf{u}$. In this regard, we present the following results that characterize the classical strategies for any n -party nonlocal game, as follows.

Proposition 15. *In any n -party nonlocal game, there exist 2^n many unique strategies that satisfy a given input-output pair $\mathbf{x} \rightarrow \mathbf{a}$.*

Proof. Let's consider the i^{th} player ($1 \leq i \leq n$) receiving an input $x_i \in \{0, 1\}$ and sending back the output $a_i \in \{0, 1\}$. When $x_i = a_i$, both a constant strategy $s_i = a_i \in \{0, 1\}$ and an input-dependent strategy $s_i = x_i$ will yield the same output. Similarly,

when $x_i \neq a_i$, the constant strategy $s_i = a_i \in \{0, 1\}$ and the input-dependent strategy $s_i = \bar{x}_i$ will produce the same output. Consequently, for each player, there exist two unique strategies resulting in the same output. Therefore, the total number of unique strategies for all n players combined is 2^n . \square

Next, we extend Theorem 15 to any n -party nonlocal games where the players can not produce a common winning strategy for given input-output pairs.

Theorem 16. *In an n -party nonlocal game, if the input bits are identical at the i^{th} position but the corresponding output bits differ, no classical strategy can simultaneously satisfy both outputs.*

Proof. Let $\mathbf{x} = x_1, x_2, \dots, x_n$ be the input bit string and $\mathbf{a} = a_1, a_2, \dots, a_n$ be the corresponding winning output. Similarly, for the input bit string $\mathbf{y} = y_1, y_2, \dots, y_n$, the winning output is $\mathbf{b} = b_1, b_2, \dots, b_n$.

$$\mathbf{x} \rightarrow \mathbf{a}, \text{ and } \mathbf{y} \rightarrow \mathbf{b}.$$

From the assumption of the theorem, $x_i = y_i$ and $a_i \neq b_i$. We prove this by contradiction. Suppose, the strategy $\mathbf{s} = s_1, s_2, \dots, s_n$ satisfy both the outputs simultaneously. Now, either $s_i \in \{0, 1\}$ or $s_i \in \{x_i, \bar{x}_i\}$.

Case 1: $s_i \in \{0, 1\}$. This implies $a_i = b_i$, which is a contradiction.

Case 2: $s_i \in \{x_i, \bar{x}_i\}$. WLOG, $s_i = x_i$. Since $x_i = y_i$, we have $a_i = x_i = y_i = b_i$, which is again a contradiction. Hence, there is no classical strategy \mathbf{s} that can simultaneously satisfy both the outputs. \square

The output bit-strings, for which there is no shared classical strategy, are called inconsistent outputs. The maximum classical winning probability in an n -party nonlocal game decreases as inconsistent output bit-strings increase. In this regard, we have the following corollary.

Corollary 2. *For all $n \in \mathbb{N}$, there exists an n -party nonlocal game for which the maximum classical winning probability is given by $1/2^{n-1}$.*

Corollary 2 identifies a class of n -party nonlocal games where the classical winning probability is very low, suggesting that exploring quantum advantages might be more feasible in this class of games. Moreover, for every binary input - binary output

n -party nonlocal game the maximum success probability corresponding to 4^n many classical strategies must belong to the set $\{\frac{k}{2^n} : k \in \{2, 3, \dots, 2^n\}\}$. This is from the fact that, there are 2^n many possible input bit-strings, among which at least two of them is guaranteed to have a common shared strategy (see Theorem 22).

7.4 Some Basic Results

In this section, we derive some basic results which are necessary throughout this chapter. It is clear from the group of strategies that for a particular input, the four different strategies of a particular group provide four different outputs. However the two different strategies from two different groups may collide, i.e., may generate the same output for a particular input. For example, the mixed strategy $x0$ (belongs to group 3) and the dependent strategy xy (belongs to group 2) both generate the output 10 for the input 10. But the two strategies which provide the same output for a particular input may not provide the same output for any other inputs. For example, the constant strategy 00 and the dependent strategy xy always provide the same output (i.e., the output 00) for the input 00 but these two strategies always provide two different outputs for all the other inputs. Some interesting results (which are required for further analysis) related to these strategies and the groups are mentioned here. We also extend some results from two-party nonlocal games to n -party nonlocal games in a more structured form.

Theorem 17. *If two different strategies either one from the constant group and the other from dependent group or one from the first mixed group (i.e., group 3) and the other from the second mixed group (i.e., group 4) provide same output for a particular input then these two strategies must provide two different outputs for all the other inputs.*

Proof. : Here the two different strategies are either from constant and dependent groups or from the two mixed groups.

Case 1: For every input, there exists a constant and a dependent strategy that provides the same output. Now, whenever the input changes, the constant strategy always provides the same output as before. However the dependent strategy provides different output than the previous one as the output of a dependent strategy always

depends on the inputs and provides different outcomes for different inputs.

Case 2: For every input, there exists a strategy from the first mixed group and another strategy from the second mixed group which provides the same output. In a mixed strategy, one bit of the output is constant and the other bit of the output is input-dependent. So for the first mixed group, there are two types of strategies, namely, xc_1 and $\bar{x}c_1$ and for the second mixed group, there are two types of strategies, namely, c_2y and $c_2\bar{y}$ where c_1 and c_2 denote the constant bits and x and y denote the corresponding dependent bits. This implies that there can be four different choices for the pair of strategies that provide the same output.

Let us first consider the case where xc_1 and c_2y be the two strategies which provide same output for the input xy . Then the corresponding outputs are,

$$xy \rightarrow xc_1 \quad (\text{applying strategy } xc_1)$$

$$xy \rightarrow c_2y \quad (\text{applying strategy } c_2y)$$

This two strategies provides same output for this input i.e., $xc_1 = c_2y$.

Now whenever these two strategies xc_1 and c_2y are applied to the input $\bar{x}y$, the corresponding outputs are,

$$\bar{x}y \rightarrow \bar{x}c_1 \quad (\text{applying strategy } xc_1)$$

$$\bar{x}y \rightarrow c_2y \quad (\text{applying strategy } c_2y)$$

As $xc_1 = c_2y$, $\bar{x}c_1 \neq c_2y$. So the two outputs are different.

Similarly whenever this two strategies are applied to the input $x\bar{y}$, the corresponding outputs are,

$$x\bar{y} \rightarrow xc_1 \quad (\text{applying strategy } xc_1)$$

$$x\bar{y} \rightarrow c_2\bar{y} \quad (\text{applying strategy } c_2y)$$

As $xc_1 = c_2y$, $c_2\bar{y} \neq xc_1$. So the two outputs are different.

Similarly whenever this two strategies are applied to the input \overline{xy} , the corresponding outputs are,

$$\overline{xy} \rightarrow \bar{x}c_1 \quad (\text{applying strategy } xc_1)$$

$$\overline{xy} \rightarrow c_2\overline{y} \quad (\text{applying strategy } c_2y)$$

As $xc_1 = c_2y$, $\overline{xc_1} = \overline{c_2y} \neq c_2\overline{y}$. So the two outputs are different.

In this similar way, one can also argue the cases for other pair of strategies. \square

Now we extend this theorem to any n -party nonlocal games in a more structured form.

Theorem 18. *In an n -party nonlocal game, if two strategies \mathbf{s} and \mathbf{t} produce the same output for a specific input bit-string \mathbf{x} , where $s_i \in \{0, 1\} \Rightarrow t_i \in \{x_i, \overline{x_i}\}$ and vice versa for all $1 \leq i \leq n$, then these two strategies must produce different outputs for all other input strings provided by the referee.*

Proof. We proof this by contradiction. Let \mathbf{y} be another input different from \mathbf{x} , and applying strategies \mathbf{s} and \mathbf{t} on \mathbf{y} yields the outputs \mathbf{a} and \mathbf{b} respectively, where $\mathbf{a} = \mathbf{b}$. Additionally, let \mathbf{x} and \mathbf{y} differ at the i^{th} position, i.e., $x_i \neq y_i = \overline{x_i}$. That is, $\mathbf{y} \xrightarrow{\mathbf{s}} \mathbf{a}$, and $\mathbf{y} \xrightarrow{\mathbf{t}} \mathbf{b}$. Then,

Case 1: $s_i \in \{0, 1\}$ and $t_i \in \{x_i, \overline{x_i}\}$. Then, either $a_i = x_i$ (when $t_i = x_i$) or $a_i = \overline{x_i}$ (when $t_i = \overline{x_i}$). If $a_i = x_i$, then $b_i = y_i$. Since $x_i \neq y_i$, it follows that $a_i \neq b_i$, implying $\mathbf{a} \neq \mathbf{b}$. Similarly, if $a_i = \overline{x_i}$, then $b_i = \overline{y_i}$. Since $\overline{x_i} \neq \overline{y_i}$, we again have $a_i \neq b_i$ and thus $\mathbf{a} \neq \mathbf{b}$, which is a contradiction.

Case 2: $s_i \in \{x_i, \overline{x_i}\}$ and $t_i \in \{0, 1\}$. In a similar manner, either $b_i = x_i$ (when $s_i = x_i$) or $b_i = \overline{x_i}$ (when $s_i = \overline{x_i}$). If $b_i = x_i$, then $a_i = y_i$. Since $y_i \neq x_i$, it follows that $a_i \neq b_i$, implying $\mathbf{a} \neq \mathbf{b}$. Similarly, if $b_i = \overline{x_i}$, then $a_i = \overline{y_i}$. Since $\overline{y_i} \neq \overline{x_i}$, we again have $a_i \neq b_i$ and thus $\mathbf{a} \neq \mathbf{b}$, which is again a contradiction.

Hence, the outputs produced by these two strategies corresponding to any other input must be different. \square

Theorem 19. *If a pair of strategies from two distinct groups provide the same output for a particular input xy then this pair of strategies must provide two different outputs for the complement input \overline{xy} .*

Proof. : Here xy is the input for which two different strategies from two different groups provide the same output. From the result of theorem 17, it can be easily argued that if the two strategies are from constant and dependent groups or from the

two mixed groups then these two strategies must provide two different outputs for the complement input \overline{xy} .

So there are two remaining cases that may occur. The first case is that whenever one strategy is from the constant group and the other strategy is from any one of the two mixed groups and the second case is that whenever one strategy is from the dependent group and the other strategy is from any one of the two mixed groups.

Case 1: In this case, one strategy from the constant group and the other strategy from one of the two mixed groups provide the same output (say ab) for the input xy . Whenever these two strategies are applied to the complement input \overline{xy} , then one can easily check that the constant strategy provides the output ab but the mixed strategy provides the output either \overline{ab} or $a\overline{b}$.

Case 2: Similarly in this case, if one strategy from the dependent group and the other strategy is from one of the two mixed groups provide the same output (say ab) for the input xy , then the dependent strategy provides the output \overline{ab} and the mixed strategy provides the output either \overline{ab} or $a\overline{b}$ for the complement input \overline{xy} . This proves the result. \square

Corollary 3. *From the results of theorem 17 and theorem 19, one can conclude that whenever there are two strategies in which one is from the constant (dependent) group and the other is from any one of the two mixed groups provide the same output for the input xy , then these strategies may not always provide two different outputs for the input $x\overline{y}$ and $\overline{x}y$.*

For example, the dependent strategy $x\overline{y}$ and the mixed strategy $0\overline{y}$ both provide the output 00 for the input 01 however these two strategies also provide the output 01 for the input 00. So whenever the inputs are not complement to each other, one can't conclude anything about the outcomes.

Theorem 20. *In an n -party nonlocal game, if a pair of strategies \mathbf{s} and \mathbf{t} yield the same output for a particular input \mathbf{x} , then the same pair of strategies must produce two different outputs for the complement input $\overline{\mathbf{x}}$.*

Proof. Since \mathbf{s} and \mathbf{t} are distinct strategies, there exists at least one index i ($1 \leq i \leq n$) such that $s_i \in \{0, 1\}$ and $t_i \in \{x_i, \overline{x}_i\}$. Let us assume that the output corresponding

to input x_i using strategy s_i is a_i , which implies that the output corresponding to input \bar{x}_i would also be a_i . $\left(x_i \xrightarrow{s_i} a_i \Rightarrow \bar{x}_i \xrightarrow{s_i} a_i\right)$.

As both strategies \mathbf{s} and \mathbf{t} produce the same output for \mathbf{x} , one can conclude that $a_i = t_i (\in \{x_i, \bar{x}_i\})$.

Case 1: $t_i = x_i$. The output for \bar{x}_i with strategy t_i is same as $\bar{x}_i (= \bar{t}_i \neq a_i)$. Hence, the corresponding outputs for $\bar{\mathbf{x}}$ are different. $\bar{x}_i \xrightarrow{s_i} x_i \Rightarrow \bar{x}_i \xrightarrow{t_i} \bar{x}_i$.

Case 2: $t_i = \bar{x}_i$. The output corresponding to \bar{x}_i using strategy t_i is $x_i (= \bar{t}_i \neq a_i)$. Again the corresponding outputs for $\bar{\mathbf{x}}$ are different. $\bar{x}_i \xrightarrow{s_i} \bar{x}_i \Rightarrow \bar{x}_i \xrightarrow{t_i} x_i$.

Hence, the outputs corresponding to $\bar{\mathbf{x}}$ on strategies \mathbf{s} and \mathbf{t} must be distinct. \square

Theorem 20 is a direct extension of Theorem 19 from two-party CHSH-like set-up to any n -party nonlocal games.

Theorem 21. *For a complement input pair (i.e., for two inputs of the form xy and $\bar{x}\bar{y}$), if one input has m many outputs and the other input has n many outputs then there are exactly mn many strategies such that each of them satisfies an output for both the inputs.*

Proof. : For a complement input pair we show that if each input has exactly one valid output, then there is exactly one strategy that satisfies both inputs. Let us consider that the input xy has output ab and input $\bar{x}\bar{y}$ has any one of the four outcomes $ab, \bar{a}b, a\bar{b}$ and $\bar{a}\bar{b}$.

One can easily check that whenever $\bar{x}\bar{y}$ has output ab then the common strategy is a constant strategy, whenever $\bar{x}\bar{y}$ has output either $\bar{a}b$ or $a\bar{b}$ then the common strategy is a mixed strategy and whenever $\bar{x}\bar{y}$ has output $\bar{a}\bar{b}$ then the common strategy is an input-dependent strategy which satisfies the outputs for both the inputs xy and $\bar{x}\bar{y}$.

This implies that there must be a strategy corresponding to every different output pair for two complement inputs. So for a complement input pair, if one input has m many outcomes and the other input has n many outcomes, then there are exactly mn different pair of outcomes. Moreover for each pair of outcomes, there exist a strategy that satisfies the outcomes for both the inputs. So, there are exactly mn many strategies which satisfy an output for both the inputs. \square

Lemma 8. *In an n -party nonlocal game, if the output corresponding to input \mathbf{x} is \mathbf{a} and the output corresponding to input $\bar{\mathbf{x}}$ is \mathbf{b} , then there exists a common strategy that satisfy both the input-output pairs, simultaneously.*

Proof. We prove this by construction. Let $\mathbf{s} = s_1, \dots, s_n$ be a strategy. For each index i ($1 \leq i \leq n$), if $a_i = b_i \in \{0, 1\}$, we choose $s_i = a_i \in \{0, 1\}$. If $a_i \neq b_i$, two possible scenarios can arise: either $a_i = x_i$ and $b_i = \bar{x}_i$, in which we can choose $s_i = x_i$, or $a_i = \bar{x}_i$ and $b_i = x_i$, in which case we can choose $s_i = \bar{x}_i$. \square

Theorem 22. *Given a complement input pairs \mathbf{x} and $\bar{\mathbf{x}}$, if one has m many output strategies, and its complement has n many output strategies, then there are exactly mn (m times n) many strategies that satisfy both the outputs.*

Proof. Based on Lemma 8, the outputs of complement input pairs can be represented using a common strategy. Moreover, since a single strategy can not produce two different output bit-strings for a single input bit-string, the uniqueness of each of these shared strategies are satisfied.

Therefore, when an input bit-string has m many winning strategies, and its complement input has n many winning strategies, there are exactly mn many strategies that satisfy both the outputs. \square

Theorem 22 is a direct extension of Theorem 21 from two-party nonlocal games to any n -party nonlocal games.

Theorem 23. *If an input (say xy) has a complement output pair and the corresponding complement input (i.e., \bar{xy}) also has two outcomes (may not be complement), then the four strategies corresponding to this input pair xy, \bar{xy} must provide four different outcomes for atleast one of the rest two inputs (i.e., for inputs $\bar{x}y$ and $x\bar{y}$).*

Proof. : Whenever each of the inputs of the complement input pair xy, \bar{xy} has two outcomes and the input xy has a complement output pair, the input \bar{xy} has two possibilities for output. Either the outcomes of \bar{xy} are complement to each other or they are not complement to each other.

Case 1: Whenever the input \bar{xy} has complement output pair, then also there are two possibilities. Either \bar{xy} has the same complement pair as in xy or the complement pair of \bar{xy} is different from the output of xy .

Whenever xy and \overline{xy} have the same complement output pair, one can check that the common strategies are 2 constant and 2 dependent strategies. We can easily verify that for the input pairs xy, \overline{xy} whenever a constant and a dependent strategy collide for a particular input, the same constant and dependent strategy must not collide for the other input (rather the same constant strategy collide with the other dependent strategy for the other input). From the result of theorem 17, we can argue that for this case, the four common strategies must provide four different outputs for all the rest two inputs. Similarly, one can conclude this same result for the case when \overline{xy} has a different complement output pair than xy .

Case 2: Whenever the input \overline{xy} has non-complement output pair, then the common strategies are one constant, one input-dependent and two mixed strategies for xy, \overline{xy} pair. One can verify that among two mixed strategies, one collides with the constant strategy and the other collides with the dependent strategy for input xy . But for input \overline{xy} , the two mixed strategies and the constant and the dependent strategy collide among themselves. Now for any one of the remaining two inputs, the constant strategy collides with those mixed strategies for which they provide different outputs for the input xy and similarly for the dependent and other mixed strategies. So from the result of theorem 17 and theorem 19, one can conclude that these four strategies must provide four different outputs for the remaining input. This concludes the proof. \square

Theorem 24. *In an n -party nonlocal game, if an input bit-string (\mathbf{x}) and its complement ($\overline{\mathbf{x}}$) has complement output pairs $\mathbf{a}, \overline{\mathbf{a}}$ and $\mathbf{b}, \overline{\mathbf{b}}$, respectively, then the four shared strategies corresponding to input bit-strings \mathbf{x} and $\overline{\mathbf{x}}$ must yield four distinct outputs for all other input bit-strings.*

Proof. Suppose, a strategy $\mathbf{s} = s_1, s_2, \dots, s_n$ yields the output \mathbf{a} for input \mathbf{x} and output \mathbf{b} for input $\overline{\mathbf{x}}$, and another strategy $\mathbf{t} = t_1, t_2, \dots, t_n$ yields the output $\overline{\mathbf{a}}$ for input \mathbf{x} and \mathbf{b} for input $\overline{\mathbf{x}}$.

$$\mathbf{x} \xrightarrow{\mathbf{s}} \mathbf{a}, \overline{\mathbf{x}} \xrightarrow{\mathbf{s}} \mathbf{b}, \text{ and } \mathbf{x} \xrightarrow{\mathbf{t}} \overline{\mathbf{a}}, \overline{\mathbf{x}} \xrightarrow{\mathbf{t}} \mathbf{b}.$$

Then, we claim, $s_i \in \{0, 1\} \Rightarrow t_i \in \{x_i, \overline{x_i}\}$ and vice versa for all $1 \leq i \leq n$.

Case 1: $s_i \in \{0, 1\}$. This implies, $a_i = b_i (\neq \overline{a_i})$. Therefore the strategy t_i produce different outputs for $\overline{x_i}$ and x_i , concluding $t_i \in \{x_i, \overline{x_i}\}$.

Case 2: $s_i \in \{x_i, \bar{x}_i\}$. This implies $a_i \neq b_i \Rightarrow b_i = \bar{a}_i$. Since, t_i yields \bar{a}_i for input x_i and b_i for input \bar{x}_i , and $\bar{a}_i = b_i$, we have $t_i \in \{0, 1\}$.

Since, the shared strategies \mathbf{s} and \mathbf{t} producing identical output for the input $\bar{\mathbf{x}}$ satisfy $s_i \in \{0, 1\} \Rightarrow t_i \in \{x_i, \bar{x}_i\}$ and vice versa for all $1 \leq i \leq n$, from Theorem 18 we can conclude that the strategies \mathbf{s} and \mathbf{t} will produce distinct outputs for all other input bit-strings.

Next, we prove the uniqueness of the output bit-string generated by each shared strategy. Consider another shared strategy $\mathbf{u} = u_1, u_2, \dots, u_n$ that yields the output \mathbf{a} for input \mathbf{x} and output $\bar{\mathbf{b}}$ for input $\bar{\mathbf{x}}$. $(\mathbf{x} \xrightarrow{\mathbf{u}} \mathbf{a}, \bar{\mathbf{x}} \xrightarrow{\mathbf{u}} \bar{\mathbf{b}})$.

Since \mathbf{s} and \mathbf{u} yield similar output as the strategies \mathbf{s} and \mathbf{t} , we can deduce that $s_i \in \{0, 1\} \Rightarrow u_i \in \{x_i, \bar{x}_i\}$ and vice versa for all $1 \leq i \leq n$. Therefore, $t_i \in \{y_i, \bar{y}_i\} \Rightarrow u_i \in \{y_i, \bar{y}_i\}$ for all $1 \leq i \leq n$.

As \mathbf{s} and \mathbf{u} satisfy the assumption of Theorem 18 and produce the same output for \mathbf{x} , they must yield different outputs for all other input bit-strings (from Theorem 18).

Furthermore, since \mathbf{t} and \mathbf{u} yield different outputs for both \mathbf{x} and $\bar{\mathbf{x}}$, and $t_i \in \{y_i, \bar{y}_i\} \Rightarrow u_i \in \{y_i, \bar{y}_i\}$ for all $1 \leq i \leq n$, it follows that $t_i = \bar{u}_i$ for all $1 \leq i \leq n$. Therefore, \mathbf{u} and \mathbf{t} cannot produce the same output for any other input bit-string.

Hence, the output bit-strings produced by each shared strategy are distinct, which implies, the four shared strategies correspond to input bit-strings \mathbf{x} and $\bar{\mathbf{x}}$ must yield four distinct outputs for all other input bit-strings. \square

Theorem 24 modifies Theorem 23, in a specific direction, demonstrating a class of n -party nonlocal games where classical winning strategies are restricted if both the inputs of a complement input pair have a complement output pair. On the other hand, Corollary 4 proves the existence of n -party nonlocal games where the strategies are not restricted if the assumptions of Theorem 24 partially fail to hold.

Corollary 4. *In the above theorem (Theorem 24), if an input bit-string \mathbf{x} has complement output pairs $\mathbf{a}, \bar{\mathbf{a}}$, however its complement input string $\bar{\mathbf{x}}$ does not have complement output pairs, \mathbf{b}, \mathbf{c} , then the four shared strategies corresponding to input bit-strings \mathbf{x} and $\bar{\mathbf{x}}$ may not produce four distinct outputs for all other input bit-strings.*

Proof. We can prove this by construction. Suppose, a strategy $\mathbf{s} = s_1, s_2, \dots, s_n$ yields the output \mathbf{a} for input \mathbf{x} and output \mathbf{b} for input $\bar{\mathbf{x}}$, and another strategy

$\mathbf{t} = t_1, t_2, \dots, t_n$ yields the output \mathbf{a} for input \mathbf{x} and \mathbf{c} for input $\bar{\mathbf{x}}$.

$$\mathbf{x} \xrightarrow{\mathbf{s}} \mathbf{a}, \bar{\mathbf{x}} \xrightarrow{\mathbf{s}} \mathbf{b}, \text{ and } \mathbf{x} \xrightarrow{\mathbf{t}} \mathbf{a}, \bar{\mathbf{x}} \xrightarrow{\mathbf{t}} \mathbf{c}.$$

We constructively prove the existence of an input bit-string $\mathbf{y} = y_1, y_2, \dots, y_n$, where the strategies \mathbf{s} and \mathbf{t} yield the same output bit-string.

Case 1: Both s_i and t_i are either in $\{0, 1\}$ or in $\{x_i, \bar{x}_i\}$. Since s_i and t_i produce the same output a_i for input bit x_i , we conclude that $s_i = t_i$, i.e., s_i and t_i will yield the same output irrespective of the i^{th} input bit. In such scenario, we set $y_i = \bar{x}_i$ to ensure that the input \mathbf{y} is distinct from \mathbf{x} .

Case 2: If $s_i \in \{0, 1\}$ and $t_i \in \{x_i, \bar{x}_i\}$, we assign $y_i = s_i$ when $t_i = x_i$, and $y_i = \bar{s}_i$ when $t_i = \bar{x}_i$. Similarly, when $t_i \in \{0, 1\}$ and $s_i \in \{x_i, \bar{x}_i\}$, we assign y_i following a similar approach.

In this manner, we construct an input bit-string \mathbf{y} for which the strategies \mathbf{s} and \mathbf{t} yield the same output bit-string. The constructed input bit-string \mathbf{y} differs from \mathbf{x} specifically at the index where $s_i = t_i$. Considering that \mathbf{b} and \mathbf{c} are not complements, there exists at least one index i ($1 \leq i \leq n$) where $b_i = c_i$.

$$x_i \xrightarrow{s_i} a_i \xleftarrow{t_i} x_i, \text{ and } \bar{x}_i \xrightarrow{s_i} b_i = c_i \xleftarrow{t_i} \bar{x}_i.$$

For such an index i , it is impossible for $s_i \in \{0, 1\}$ and $t_i \in \{x_i, \bar{x}_i\}$ to occur simultaneously. If $s_i \in \{0, 1\}$, then $a_i = b_i$, but $t_i \in \{x_i, \bar{x}_i\}$ cannot produce the same output for two complementary bits. Therefore, both s_i and t_i must either be in $\{0, 1\}$ or in $\{x_i, \bar{x}_i\}$, implying $s_i = t_i$.

Hence, we constructively found an input bit-string \mathbf{y} (distinct from \mathbf{x}) for which the strategies \mathbf{s} and \mathbf{t} yield the same output bit-string. \square

Lemma 9. : *If an input (say xy) has a complement output pair and its complement input (i.e., \overline{xy}) has only one outcome then the two strategies from xy, \overline{xy} pair must provide non-complement output pairs for each of the rest two inputs.*

Proof. : Let us consider that the input xy has two complement outputs of the form ab and \overline{ab} . Then the complement input \overline{xy} has two possibilities, either the output of \overline{xy} is same as one of the outputs of xy or the output of \overline{xy} is different from the output of xy .

Case 1: Whenever the output of \overline{xy} is same with one of the output of xy , there are one constant strategy and one input-dependent strategy. Let us assume that the common output of xy and \overline{xy} is ab . Then the constant strategy is ab and the input-dependent strategy is mn (say) where $m \in \{x, \overline{x}\}$ and $n \in \{y, \overline{y}\}$. This implies that,

$$\begin{aligned} xy &\rightarrow \overline{ab} \quad (\text{applying strategy } mn) \\ \overline{xy} &\rightarrow ab \quad (\text{applying strategy } mn) \end{aligned}$$

Thus we conclude that

$$\begin{aligned} x &\rightarrow \overline{a} \quad (\text{applying } m) \\ y &\rightarrow \overline{b} \quad (\text{applying } n) \end{aligned}$$

Hence it is clear that the strategy mn provides the output \overline{ab} for the input \overline{xy} and provides the output \overline{ab} for input $x\overline{y}$. So the two strategies ab and mn provide non-complement output pair ab and \overline{ab} for the input \overline{xy} and provide non-complement output pair ab and \overline{ab} for the input $x\overline{y}$.

Case 2: Similarly, whenever the output of \overline{xy} is different from the outputs of xy , there are two mixed strategies from two different groups. Let us consider that the outputs of xy are ab and \overline{ab} and the output of \overline{xy} is either \overline{ab} or \overline{ab} . Without loss of generality, we assume that the output of \overline{xy} is \overline{ab} . Let the two mixed strategies are mc_1 and c_2n , where $m \in \{x, \overline{x}\}$ and $n \in \{y, \overline{y}\}$ are the dependent bits and $c_1, c_2 \in \{0, 1\}$ are the constant bits. If we consider that the strategy mc_1 provides output ab and c_2n provides output \overline{ab} for input xy then one can easily check that,

$$\begin{aligned} x &\rightarrow a \quad (\text{applying } m) \\ y &\rightarrow \overline{b} \quad (\text{applying } n) \end{aligned}$$

It is also clear that $c_1 = b$ and $c_2 = \overline{a}$.

Hence, one can easily check that the strategy mc_1 and c_2n provide outputs \overline{ab} and \overline{ab} respectively for input \overline{xy} . Similarly, one can also check that the strategy mc_1 and c_2n provide outputs ab and \overline{ab} respectively for input $x\overline{y}$. This implies that the mixed

strategies from two different groups also provide non-complement output pair for the rest of the two inputs. Similarly, one can also check the other cases. \square

We extend Lemma 9 from two-party nonlocal games to n-party nonlocal games as follows:

Lemma 10. *In an n-party nonlocal game, if an input bit-string \mathbf{x} has complement output pairs $\mathbf{a}, \bar{\mathbf{a}}$, and its complement input string $\bar{\mathbf{x}}$ has an output \mathbf{b} , then the two shared strategies corresponding to \mathbf{x} and $\bar{\mathbf{x}}$ can not produce two complementary outputs for any other input bit-string.*

Proof. Suppose $\mathbf{s} = s_1, \dots, s_n$ and $\mathbf{t} = t_1, \dots, t_n$ be the shared strategies corresponding to the inputs \mathbf{x} and $\bar{\mathbf{x}}$. $(\mathbf{x} \xrightarrow{\mathbf{s}} \mathbf{a}, \mathbf{x} \xrightarrow{\mathbf{t}} \bar{\mathbf{a}}, \text{ and } \bar{\mathbf{x}} \xrightarrow{\mathbf{s}} \mathbf{b} \xleftarrow{\mathbf{t}} \bar{\mathbf{x}})$.

From Theorem 24, we know $s_i \in \{0, 1\} \Rightarrow t_i \in \{x_i, \bar{x}_i\}$ and vice-versa for all $i (1 \leq i \leq n)$. Also, from Theorem 18, \mathbf{s} and \mathbf{t} yields two distinct outputs for all other input bit-string. Let \mathbf{y} be any input bit-string other than \mathbf{x} such that $\mathbf{y}\mathbf{y}$ yields output $\mathbf{c} = c_1, \dots, c_n$ on strategy \mathbf{s} and output $\mathbf{d} = d_1, \dots, d_n$ on strategy \mathbf{t} . $(\mathbf{y} \xrightarrow{\mathbf{s}} \mathbf{c}, \mathbf{y} \xrightarrow{\mathbf{t}} \mathbf{d})$. The goal is to show $c_i = d_i$ for atleast one index position i .

Since, \mathbf{y} is different from \mathbf{x} , there exists atleast one index point i such that $y_i = \bar{x}_i$. Since, $\bar{x}_i \xrightarrow{s_i} b_i \xleftarrow{t_i} \bar{x}_i$, for y_i , the output bit is $b_i = c_i = d_i$. Hence, \mathbf{c} and \mathbf{d} can not be complement to each other. \square

7.5 Analysis of the Maximum Success Probability in Classical Scenario

In the classical scenario of a nonlocal game, the players have to fix some strategies before the game begins because after getting the input bits from the referee, the players aren't allowed to communicate with each other. For the binary input binary output two-party nonlocal games, each player has only two possibilities for their input bits (either 0 or 1) and has two choices (either 0 or 1) for the output bits. In this scenario, each player has atmost 4 different strategies (either output 0 or 1 or the input bit itself or the complement of the input bit) to generate their output bits. This implies that for a particular two-bit input string provided by the referee, there are atmost 16 different strategies for the two players in a classical scenario.

Among these 16 different strategies, the strategy which provides the maximum success probability for a particular game is the optimal classical strategy and the corresponding success probability is the maximum classical success probability for this game. All the possible output strategies (by the two players) for a particular game and their corresponding success probabilities can be represented in a tabular form as mentioned in Table 7.2.

In this Table, each p_i denotes the fraction of inputs for which the game can be won using the i -th strategy. The two players can choose any of these 16 different strategies before the game begins and later can output their bits accordingly. For example, if they choose the first strategy specified in Table 7.2, both of them output 0 irrespective of their inputs. Similarly, whenever they choose the second strategy, the first player always outputs 0 irrespective of his inputs whereas, the second player outputs his corresponding input bit itself i.e., if he receives the input 0, he outputs 0, otherwise he outputs 1. After their output, the referee checks the fraction of inputs for which the two players win the game. The strategy which generates the winning outcomes for most of the inputs of a particular game is considered as the optimal strategy corresponding to that game. This implies that from Table 7.2, one can obtain the maximum classical success probability (p_{max}) as $p_{max} = \max_i p_i$.

As every binary input binary output two-party nonlocal game has 4 possible inputs, one can easily check that the classical success probability for each of the possible 16 strategies must belong to the set $\{0, 0.25, 0.5, 0.75, 1\}$. From the result of theorem 21, it is clear that there must be a classical strategy corresponding to each of the complement input pairs. This implies that for every binary input binary output two-party nonlocal game, the maximum classical success probability must be at least 0.5. Similarly, if a game has inconsistent outputs (as mentioned in subsection 7.2) for an input pair, then according to the discussion of subsection 7.2, the maximum classical success probability must be less than 1 (i.e., either 0.5 or 0.75).

7.6 Analysis of the Maximum Success Probability in Quantum Scenario

In the quantum strategy of a two-party nonlocal game, the two players initially share some entanglement among themselves (before the game begins) and then during the

Output for Alice (a)		Output for Bob (b)		Success Probability
Output for input $x = 0$	Output for input $x = 1$	Output for input $y = 0$	Output for input $y = 1$	
0	0	0	0	p_1
0	0	0	1	p_2
0	0	1	0	p_3
0	0	1	1	p_4
0	1	0	0	p_5
0	1	0	1	p_6
0	1	1	0	p_7
0	1	1	1	p_8
1	0	0	0	p_9
1	0	0	1	p_{10}
1	0	1	0	p_{11}
1	0	1	1	p_{12}
1	1	0	0	p_{13}
1	1	0	1	p_{14}
1	1	1	0	p_{15}
1	1	1	1	p_{16}

Table 7.2: Success probabilities of a game for all possible classical strategies

game, they perform some specific (unitary) operations on their qubits (based on the inputs) and measure their qubits to get the output bits.

Let us assume that the two players (say Alice and Bob) share the Bell-state $|\psi\rangle_{AB} = \frac{|00\rangle + |11\rangle}{\sqrt{2}}$ among themselves and Alice measures in $\theta_0(\theta_1)$ rotated basis for the input 0(1) and Bob measures in $\psi_0(\psi_1)$ rotated basis for the input 0(1).

Now, it is easy to check that whenever the referee provides the input bit 0 to both Alice and Bob (i.e., for the input 00), the shared states between Alice and Bob (after applying their respective unitary operators) are of the form

$$\begin{aligned}
& \frac{1}{\sqrt{2}} [(\cos \theta_0|0\rangle + \sin \theta_0|1\rangle)(\cos \psi_0|0\rangle + \sin \psi_0|1\rangle)] \\
& + \frac{1}{\sqrt{2}} [(-\sin \theta_0|0\rangle + \cos \theta_0|1\rangle)(-\sin \psi_0|0\rangle + \cos \psi_0|1\rangle)] \\
= & \frac{1}{\sqrt{2}} [(\cos \theta_0 \cos \psi_0 + \sin \theta_0 \sin \psi_0)|00\rangle + (\cos \theta_0 \sin \psi_0 - \sin \theta_0 \cos \psi_0)|01\rangle] \\
& + \frac{1}{\sqrt{2}} [(\sin \theta_0 \cos \psi_0 - \cos \theta_0 \sin \psi_0)|10\rangle + (\cos \theta_0 \cos \psi_0 + \sin \theta_0 \sin \psi_0)|11\rangle] \\
= & \frac{1}{\sqrt{2}} [\cos(\theta_0 - \psi_0)|00\rangle - \sin(\theta_0 - \psi_0)|01\rangle + \sin(\theta_0 - \psi_0)|10\rangle + \cos(\theta_0 - \psi_0)|11\rangle]
\end{aligned}$$

So for the input 00, the probability of getting each of the outputs 00 and 11 is $\frac{1}{2} \cos^2(\theta_0 - \psi_0)$ and the probability of getting each of the outputs 01 and 10 is $\frac{1}{2} \sin^2(\theta_0 - \psi_0)$.

Similarly for the input 01, the shared states between Alice and Bob after applying the specific unitary operations are of the form $\cos(\theta_0 - \psi_1)|00\rangle - \sin(\theta_0 - \psi_1)|01\rangle + \sin(\theta_0 - \psi_1)|10\rangle + \cos(\theta_0 - \psi_1)|11\rangle$. So in this case, the probability of getting each of the outputs 00 and 11 is $\frac{1}{2} \cos^2(\theta_0 - \psi_1)$ and the probability of getting each of the outputs 01 and 10 is $\frac{1}{2} \sin^2(\theta_0 - \psi_1)$.

In this similar way, one can easily check that for the input 10, the shared states between Alice and Bob after applying the specific unitaries are of the form $\cos(\theta_1 - \psi_0)|00\rangle - \sin(\theta_1 - \psi_0)|01\rangle + \sin(\theta_1 - \psi_0)|10\rangle + \cos(\theta_1 - \psi_0)|11\rangle$ and the corresponding probabilities are $\frac{1}{2} \cos^2(\theta_1 - \psi_0)$ (for each of the outputs 00 and 11) and $\frac{1}{2} \sin^2(\theta_1 - \psi_0)$ (for each of the outputs 01 and 10).

Similarly for the input 11, the shared states between Alice and Bob after applying the specific unitaries are of the form $\cos(\theta_1 - \psi_1)|00\rangle - \sin(\theta_1 - \psi_1)|01\rangle + \sin(\theta_1 - \psi_1)|10\rangle + \cos(\theta_1 - \psi_1)|11\rangle$ and the corresponding probabilities are $\frac{1}{2} \cos^2(\theta_1 - \psi_1)$ (for each of the outputs 00 and 11) and $\frac{1}{2} \sin^2(\theta_1 - \psi_1)$ (for each of the outputs 01 and 10).

From these quantum success probability expressions for different inputs, it is clear that for a particular input xy , the probability of getting each of the outputs 00 and 11 is $\frac{1}{2} \cos^2 \alpha$ and the probability of getting each of the outputs 01 and 10 is $\frac{1}{2} \sin^2 \alpha$ where $\alpha = (\theta_x - \psi_y)$ (according to our mentioned strategy).

As for every nonlocal game, the referee is supposed to provide the input bits randomly, the two players calculate the overall success probability considering each of the inputs equally likely. For any particular game, the expression of the quantum success probability depends on the distribution of the successful outcomes for all the possible inputs. Depending on this distribution, the quantum success probability expressions involve the variables $\theta_0, \theta_1, \psi_0, \psi_1$. After getting the quantum success probability expression for a particular game, one can easily find the values of $\theta_0, \theta_1, \psi_0, \psi_1$ for which the success probability becomes maximum.

One can verify that for the games having inconsistent outputs, the quantum success probabilities corresponding to the inconsistent input pair are of the form $\frac{1}{2} \cos^2 \alpha$ and $\frac{1}{2} \sin^2 \alpha$. So, the maximum quantum success probability for these games having

inconsistent outputs will be 0.75. This implies that all the nonlocal games having inconsistent outputs may not offer a quantum advantage (i.e., the maximum quantum success probability is greater than the maximum classical success probability)¹.

7.7 Analysis of the Results in [DPM19]

It is evident from the discussion till now that every binary input binary output two-party nonlocal game can be represented as a 4-variable Boolean function. One can also consider the inputs and the outputs separately as 2-variable Boolean functions for binary input binary output two-party nonlocal games and compose these two functions to construct 4-variable functions. For example in the CHSH game, the input function is $f(x, y) = x \wedge y$ and the output function is $g(a, b) = a \oplus b$. The actual function that represents the CHSH game is just the composition of these two (input and output) functions.

Recently some analysis has been done in this direction in [DPM19] (considering all the non-constant 2-variable Boolean functions and composing every possible pairs among them to construct the corresponding 4-variable Boolean functions) to explore the performance of some 4-variable Boolean functions (or binary input binary output two-party nonlocal games) as distinguishers for the certification of different dimensional quantum states. As the authors consider only non-constant Boolean functions in [DPM19], the total number of 4-variable Boolean functions that they have considered are $(2^{2^2} - 2) \times (2^{2^2} - 2) = 14 \times 14 = 196$. However, there are total $2^{2^4} = 65536$ possible 4-variable Boolean functions. So in [DPM19], only a small fraction of the games are explored from the class of all possible binary input binary output two-party nonlocal games.

There are some miscalculations in [DPM19, Table 1] regarding the number of different Boolean functions which we would like to point out here. In Table 1 of [DPM19], it is mentioned that the total number of function pairs (f, g_2) such that each of $f(x, y)$ and $g_2(a, b)$ contains one 0(1) is 32(32). However one can verify that the total number of such function pairs is actually 16. For a detail analysis corresponding to this result, one may refer to Proposition 16.

¹In this context one should remember that every classical strategy is also a quantum one where no entanglement is shared between the parties

Proposition 16. *Let $g_i : \mathbb{Z}_2 \times \mathbb{Z}_2 \rightarrow \mathbb{Z}_2$ be a Boolean function such that $|g_i^{-1}(0)| = 1$ where $g_i^{-1}(0) = \{(x, y) \in \mathbb{Z}_2 \times \mathbb{Z}_2 : g_i(x, y) = 0\}$ for $i = 1, 2$. Let $f(x_1, x_2, x_3, x_4) = g_1(x_1, x_2) * g_2(x_3, x_4)$ where $*$ is a binary operation on \mathbb{Z}_2 . Then given a binary operation $*$, there are atmost 16 different possibilities for f .*

Proof. : Since $|g_i^{-1}(0)| = 1$, there is (a_i, b_i) such that $g_i(a_i, b_i) = 0$ and $g_i(x, y) = 1$ for $(x, y) \neq (a_i, b_i)$. Now, for each of the two functions g_1 and g_2 , there are four different choices that satisfy the above condition namely,

$$g_1^{(1)}(0, 0) = 0 \text{ and } g_1^{(1)}(x, y) = 1 \text{ for } (x, y) \neq (0, 0) \quad (7.1)$$

$$g_1^{(2)}(0, 1) = 0 \text{ and } g_1^{(2)}(x, y) = 1 \text{ for } (x, y) \neq (0, 1) \quad (7.2)$$

$$g_1^{(3)}(1, 0) = 0 \text{ and } g_1^{(3)}(x, y) = 1 \text{ for } (x, y) \neq (1, 0) \quad (7.3)$$

$$g_1^{(4)}(1, 1) = 0 \text{ and } g_1^{(4)}(x, y) = 1 \text{ for } (x, y) \neq (1, 1). \quad (7.4)$$

Similarly for g_2 , there are also four different choices namely,

$$g_2^{(1)}(0, 0) = 0 \text{ and } g_2^{(1)}(x, y) = 1 \text{ for } (x, y) \neq (0, 0) \quad (7.5)$$

$$g_2^{(2)}(0, 1) = 0 \text{ and } g_2^{(2)}(x, y) = 1 \text{ for } (x, y) \neq (0, 1) \quad (7.6)$$

$$g_2^{(3)}(1, 0) = 0 \text{ and } g_2^{(3)}(x, y) = 1 \text{ for } (x, y) \neq (1, 0) \quad (7.7)$$

$$g_2^{(4)}(1, 1) = 0 \text{ and } g_2^{(4)}(x, y) = 1 \text{ for } (x, y) \neq (1, 1). \quad (7.8)$$

Thus $f(x_1, x_2, x_3, x_4) = g_1^{(i)}(x_1, x_2) * g_2^{(j)}(x_3, x_4)$ for some $1 \leq i, j \leq 4$. As there are maximum 4 different choices for each of $g_1^{(i)}$ and $g_2^{(j)}$ for some $1 \leq i, j \leq 4$, there are atmost $4 \times 4 = 16$ different choices for f . \square

It is also mentioned (in [DPM19] Table 1) that the total number of function pairs such that $f(x, y)$ contains one 0 and $g_2(a, b)$ contains one 1 are 6. For this case also, similar to the derivation performed in the proof of proposition 16, one can verify that the total number of such function pairs is also 16.

In [DPM19], the authors have proposed the idea of distinguishing different dimensional quantum states with the help of some nonlocal games. In their paper, they have explored the performance of some two-party nonlocal games in quantum scenario with the intention of finding those games which provide significant advantage in the quantum winning probability for different dimensional states. However, the main lim-

itation in their approach is that they have explored only 196 functions from the set of 65536 possible 4-variable Boolean functions. Because of this limitation, they might not consider the game which is the most efficient as the dimensionality distinguisher (i.e., which has the maximum probability difference in the quantum scenario for different dimensional states) among all possible binary input binary output two-party nonlocal games. In this article, we have considered all those binary input binary output two-party nonlocal games which have atleast one successful outcome for every possible input and evaluate their performance both in classical and quantum scenario (considering the strategy mentioned in Subsection 7.6). However, we haven't analyzed anything regarding the performance of those games as dimensionality distinguishers.

7.8 Analysis of the Binary Input Binary Output Two-party Nonlocal Games

In the current context, we are interested in finding all those two-party binary input binary output nonlocal games where one can achieve quantum advantage. So far, CHSH game is the most well known game that offers a separation around 0.1 between the maximum classical (which is 0.75) and the maximum quantum (which is around 0.853) success probability.

From the definition of the partition introduced in definition 3, one can easily check that the CHSH game can be represented as a $2 + 2 + 2 + 2$ partition based on the distribution of its outputs. As our main intention is to find out all those games that offer quantum advantage (with maximum quantum success probability greater than the existing maximum for the two party scenario, i.e., 0.853), here we consider only those games for which the number of valid outputs corresponding to each possible input is non-zero (so that there is a chance of achieving the maximum quantum success probability greater than 0.853 for random inputs).

For every number of successful outcomes (i.e., the number of 0's in the output column of a Boolean function), we first find out all possible partitions of that outcome and then explore the performance of the games corresponding to each of those partitions to derive the maximum classical and the maximum quantum success probabilities. For example, the games having 8 successful outcomes (i.e., 8 number of 0's in the output column of the Boolean function representation of those games), there

are four possible partitions such that each input has atleast one successful outcome. In this section, we first find out all those partitions for every possible number of successful outcomes and then analyze the performance of the games corresponding to each of those partitions according to the techniques mentioned in Subsection 7.5 and Subsection 7.6.

7.8.1 Games Corresponding to 8 Successful Outcomes

In this subsection, we analyze (in details) the performance of the games corresponding to all possible partitions for 8 successful outcomes in both classical and quantum scenario.

Analysis for partition $4 + 2 + 1 + 1$:

For this partition of games, there must be a complement input pair (i.e., of the form xy and \overline{xy}) either both the inputs have 1 outcome or one input has 1 outcome and the other input has 2 outcomes. Whenever the two 1 outcomes have a complement input pair, the strategy corresponding to this input pair must satisfy one output for the input corresponding to 4 outcomes. Similarly if the 2 outcome and a 1 outcome has a complement input pair, then the two strategies corresponding to this complement input pair must satisfy atleast one output for the input having 4 outcomes.

So the minimum classical success probability for all the games of this partition is 0.75. From the discussion in subsection 7.6, one can easily check that the maximum quantum success probability corresponding to each of the inputs having 4 and 2 outcomes is 1 and for inputs having 1 outcome is 0.5. So, the maximum quantum success probability for any game of this partition is $\frac{1}{4}[1 + 1 + 0.5 + 0.5] = 0.75$. This implies that for this partition of games, one can't achieve any advantage in quantum scenario. For example, here we consider the following game corresponding to this partition. For this game, one can easily check that for the strategy $a = x$ and $b = 0$,

Input	Corresponding output
00	00, 01, 10, 11
01	00, 11
10	01
11	10

the players can win the game with probability 0.75 in classical scenario whereas in quantum scenario, the maximum quantum success probability is 0.75. This implies that for this partition of games, there is no chance of getting a quantum advantage.

Analysis for partition $3 + 3 + 1 + 1$

For this partition of games, there must be a complement input pair either they have 3 outcomes or one input has 3 and the other has 1 outcome. Let us consider that the input xy has 3 outcomes. Then for the complement input \overline{xy} , there are two possibilities, either \overline{xy} has 3 outcomes or \overline{xy} has 1 outcome.

Case 1: Whenever \overline{xy} has 3 outcomes, one can get nine strategies for xy, \overline{xy} pair. As each of the inputs xy and \overline{xy} must have a complement output pair, according to the result of theorem 23, these four strategies corresponding to these two complement output pairs must provide four different outputs for each of the rest two inputs. So one of these four strategies must satisfy atleast one output for atleast one of the rest of two inputs. Hence, the minimum classical success probability for all these games is 0.75. This implies that for this partition of games, one can't achieve any advantage in quantum scenario. For example, here we consider the following game corresponding to this partition.

Input	Corresponding output
00	00, 10, 11
01	00
10	11
11	00, 10, 11

For this game, one can easily check that for the strategy $a = 0, b = 0$ or $a = 1, b = 1$ or $a = x, b = \overline{y}$, the players can win the game with probability 0.75 in classical scenario whereas in quantum scenario, the maximum quantum success probability is 0.75. This implies that for this partition of games, there is no chance of getting a quantum advantage.

Case 2: Similarly whenever \overline{xy} has 1 valid outcome, for xy, \overline{xy} pair one can get three strategies. Among these three strategies, the two strategies that correspond to the complement output pair of xy must provide 2 different outputs for each of the rest two inputs (as mentioned in theorem 23). So one of these two strategies must

satisfy atleast one output for the rest input having 3 outcomes. Hence, the minimum classical success probability for this form of game is also 0.75. This implies that for this partition of games, one can't achieve any advantage in quantum scenario. For example, here we consider the following game corresponding to this partition.

Input	Corresponding output
00	00, 01, 11
01	00, 10, 11
10	01
11	11

For this game, one can easily check that for the strategy $a = 1, b = 1$ or $a = \bar{x}, b = \bar{y}$, the players can win the game with probability 0.75 in classical scenario.

From the discussion of subsection 7.6, one can easily verify that the maximum quantum success probability for each of the inputs having 3 outcomes is 1 and for each of the inputs having 1 outcome is 0.5. So, the maximum quantum success probability for any game of this partition (for equiprobable outcomes) is $\frac{1}{4}[1+1+0.5+0.5] = 0.75$.

It is clear from the analysis that for this partition of games, there is no chance of getting any advantage in quantum success probability as compared to the classical one.

A Game for partition $2 + 2 + 2 + 2$ having quantum advantage

For this partition of games, each of the four inputs has 2 outcomes. According to the discussion of subsections 7.5 and 7.6, one can easily check that if none of the inputs have complement output pair, the maximum quantum success probability is 0.5. Whenever 1 or 2 inputs have complement output pair (like the discussions of the previous two partitions), one can easily check that there is no advantage in quantum success probability. So to achieve quantum advantage, the games must have complement output for atleast three inputs. For the games having complement output for three inputs, one can verify that although the maximum quantum success probability is greater than 0.75, the maximum classical success probability is always 1. A well-known game of this partition having complement output pair for all the inputs is the CHSH game which offers quantum advantage. Here we consider this game and analyze its performance in both classical and quantum scenarios.

Input	Corresponding output
00	00, 11
01	00, 11
10	00, 11
11	01, 10

From the strategies mentioned in subsection 7.5, one can easily check that the maximum classical success probability for this game is 0.75 and one of the strategies to get this success probability is $a = 0$ and $b = 0$.

Similarly from the discussion of subsection 7.6, one can easily check that the expression for quantum success probability of this game is of the form

$$\begin{aligned} & \frac{1}{4} [\cos^2(\theta_0 - \psi_0) + \cos^2(\theta_0 - \psi_1) + \cos^2(\theta_1 - \psi_0) + \sin^2(\theta_1 - \psi_1)] \\ &= \frac{1}{2} + \frac{1}{8} [\cos 2\alpha + \cos 2\beta + \cos 2\gamma - \cos 2\delta] \end{aligned}$$

where $\alpha = (\theta_0 - \psi_0)$, $\beta = (\theta_0 - \psi_1)$, $\gamma = (\theta_1 - \psi_0)$ and $\delta = (\theta_1 - \psi_1)$.

Now the cosines can be written as an inner product between two unit vectors. Suppose,

$$\begin{aligned} u_0 &= \cos \theta_0 |0\rangle + \sin \theta_0 |1\rangle \\ u_1 &= \cos \theta_1 |0\rangle + \sin \theta_1 |1\rangle \\ v_0 &= \cos \psi_0 |0\rangle + \sin \psi_0 |1\rangle \\ v_1 &= \cos \psi_1 |0\rangle + \sin \psi_1 |1\rangle \end{aligned}$$

Then one can easily check that for all i, j , $u_i v_j = \cos 2(\theta_i - \psi_j)$. So one can rewrite the above expression as

$$\begin{aligned} & \frac{1}{2} + \frac{1}{8} [u_0 v_0 + u_0 v_1 + u_1 v_0 - u_1 v_1] \\ &= \frac{1}{2} + \frac{1}{8} [u_0(v_0 + v_1) + u_1(v_0 - v_1)] \\ &\leq \frac{1}{2} + \frac{1}{8} (\|v_0 + v_1\| + \|v_0 - v_1\|) \end{aligned}$$

Let us assume, $\langle v_0, v_1 \rangle = a + ib$ and $\langle v_1, v_0 \rangle = a - ib$. Then $\|v_0 + v_1\| = \sqrt{2 + 2a}$ and $\|v_0 - v_1\| = \sqrt{2 - 2a}$. It is easy to check that the expression $\sqrt{2 + 2a} + \sqrt{2 - 2a}$

attains maximum value for $a = 0$ and the corresponding maximum value is $2\sqrt{2}$ i.e., $(\|v_0 + v_1\| + \|v_0 - v_1\|) \leq 2\sqrt{2}$. So, the maximum quantum success probability for this form of games is $\left(\frac{1}{2} + \frac{2\sqrt{2}}{8}\right) = \frac{1}{2} + \frac{1}{2\sqrt{2}} \approx 0.853$.

This implies that one can find some values of $\theta_0, \theta_1, \psi_0$ and ψ_1 for which the game can be won with probability 0.853 in quantum scenario. Similarly one can show the same upper bound for some other games of this group for which the maximum classical success probability is 0.75. Therefore, the maximum separation for this partition of games is $(0.853 - 0.75) \approx 0.103$.

A Game for partition $3 + 2 + 2 + 1$ having quantum advantage

From the theoretical analysis (similar to the analysis of partition $4 + 2 + 1 + 1$ and $3 + 3 + 1 + 1$), one can easily check that not all games for this partition can be won with probability 1 in classical scenario and there are some games for which the maximum classical success probability is 0.75. From the expressions of quantum success probabilities of these games, one can easily check that for some of those games, maximum quantum success probability is greater than the classical one. Here we consider one of these games and analyze its performance in both classical and quantum scenarios. From the strategies mentioned in subsection 7.5, one can easily

Input	Corresponding output
00	00, 01, 11
01	00, 11
10	01
11	00, 11

check that the maximum classical success probability for this game is 0.75 and one of the strategies to get this success probability is $a = 0$ and $b = 0$.

Similarly from the discussion of subsection 7.6, one can easily check that the expression for quantum success probability of this game is of the form

$$\begin{aligned} & \frac{1}{4} \left[\frac{1}{2} + \frac{1}{2} \cos^2 \alpha + \cos^2 \beta + \frac{1}{2} \sin^2 \gamma + \cos^2 \delta \right] \\ &= \frac{1}{2} + \frac{1}{4} [1 + \cos 2\alpha + 2 + 2 \cos 2\beta + 1 - \cos 2\gamma + 2 + 2 \cos 2\delta] \end{aligned}$$

where $\alpha = (\theta_0 - \psi_0)$, $\beta = (\theta_0 - \psi_1)$, $\gamma = (\theta_1 - \psi_0)$ and $\delta = (\theta_1 - \psi_1)$.

One can think of the cosines as the inner products between unit vectors. In that case, one can rewrite the above as

$$\frac{1}{4} \left[2 + \frac{1}{4} (u_0 v_0 + 2u_0 v_1 - u_1 v_0 + 2u_1 v_1) \right] \leq \frac{1}{4} \left[2 + \frac{1}{4} [\|v_0 + 2v_1\| + \| -v_0 + 2v_1 \|] \right]$$

Let us assume, $\langle v_0, v_1 \rangle = a + ib$ and $\langle v_1, v_0 \rangle = a - ib$. Then $\|v_0 + 2v_1\| = \sqrt{5 + 4a}$ and $\| -v_0 + 2v_1 \| = \sqrt{5 - 4a}$. It is easy to check that the expression $\sqrt{5 + 4a} + \sqrt{5 - 4a}$ attains maximum value for $a = 0$ and the corresponding maximum value is $2\sqrt{5}$ i.e., $(\|v_0 + 2v_1\| + \| -v_0 + 2v_1 \|) \leq 2\sqrt{5}$.

Hence the maximum winning probability $\leq \frac{1}{4} [2 + \frac{1}{4} \times 2\sqrt{5}] \approx 0.78$.

This implies that one can find some values of $\theta_0, \theta_1, \psi_0$ and ψ_1 for which the game can be won with probability 0.78 in quantum scenario. Similarly one can show the same upper bound for some other games of this group for which the maximum classical success probability is 0.75. Therefore the maximum separation for this class of games is $(0.78 - 0.75) \approx 0.03$.

From this discussion, it is clear that for all the games corresponding to partitions $4 + 2 + 1 + 1$ and $3 + 3 + 1 + 1$, there are no chances of getting quantum advantage. But for the partition $2 + 2 + 2 + 2$ and $3 + 2 + 2 + 1$, there are some games which provide quantum advantage with a separation around 0.103 and 0.03 respectively. A summary of these results are mentioned in Table 7.3.

Partitions	Max. classical success prob.	Max. quantum success prob.	Corresponding Separation
4+2+1+1	0.75	0.75	NA
	1.0	1.0	NA
3+3+1+1	0.75	0.75	NA
	1.0	1.0	NA
2+2+2+2	0.75	0.853	0.103
	1.0	1.0	NA
3+2+2+1	0.75	0.78	0.03
	1.0	1.0	NA

Table 7.3: Analysis of partitions for 8 successful outcomes

7.8.2 Games Corresponding to 9 Successful Outcomes

Proceeding to the similar way as the analysis of the 8 successful outcomes, the maximum classical and quantum success probabilities that one can achieve for each of the partitions of the 9 successful outcomes are mentioned in the Table 7.4. From these results, one can easily check that quantum advantage can be achieved (with a separation around 0.042) only for some of the games corresponding to the partition $3+3+2+1$. For simplicity, here we only consider a game (having quantum advantage) from the partition $3+3+2+1$ and analyze the performance.

Partitions	Max. classical success prob.	Max. quantum success prob.	Corresponding Separation
4+3+1+1	0.75	0.75	NA
	1.0	1.0	NA
4+2+2+1	1.0	1.0	NA
3+2+2+2	1.0	1.0	NA
3+3+2+1	0.75	0.792	0.042
	1.0	1.0	NA

Table 7.4: Analysis of partitions for 9 successful outcomes

A Game for partition $3+3+2+1$ having quantum advantage

From the results of table 7.4, it is clear that for the games having 9 successful outcomes, quantum advantage can be achieved only for some of the games having partition $3+3+2+1$. Here we consider the following game which can't be won with certainty in classical scenario.

Input	Corresponding output
00	00, 11
01	00, 01, 10
10	11
11	00, 01, 11

From the strategies mentioned in subsection 7.5, one can easily check that the maximum classical success probability for this game is 0.75 and one of the strategies to get this success probability is $a = 0$ and $b = 0$.

Similarly from the discussion of subsection 7.6, one can easily check that the expression for quantum success probability of this mentioned game is of the form

$$\begin{aligned} & \frac{1}{4} \left[\cos^2 \alpha + \frac{1}{2} + \frac{1}{2} \sin^2 \beta + \frac{1}{2} + \frac{1}{2} \cos^2 \gamma + \frac{1}{2} \cos^2 \delta \right] \\ &= \frac{1}{4} \left[2 + \frac{1}{4} + \frac{1}{4} (2 \cos 2\alpha - \cos 2\beta + \cos 2\gamma + \cos 2\delta) \right] \end{aligned}$$

where $\alpha = (\theta_0 - \psi_0)$, $\beta = (\theta_0 - \psi_1)$, $\gamma = (\theta_1 - \psi_0)$ and $\delta = (\theta_1 - \psi_1)$.

One can think of the cosines as the inner products between unit vectors. In that case, one can rewrite the above as

$$\begin{aligned} & \frac{1}{4} \left[2 + \frac{1}{4} + \frac{1}{4} (2u_0v_0 - u_0v_1 + u_1v_0 + u_1v_1) \right] \\ & \leq \frac{1}{4} \left[2 + \frac{1}{4} + \frac{1}{4} (\|u_0\| \|2v_0 - v_1\| + \|u_1\| \|v_0 + v_1\|) \right] \end{aligned}$$

Let us assume that $\langle v_0, v_1 \rangle = (a + ib)$ and $\langle v_1, v_0 \rangle = (a - ib)$. Then one can easily check that $\|2v_0 - v_1\| = \sqrt{5 - 4a}$ and $\|v_0 + v_1\| = \sqrt{2 + 2a}$. From these expressions, one can easily calculate that the expression $\sqrt{2 + 2a} + \sqrt{5 - 4a}$ attains maximum value for $a = -\frac{1}{4}$ and the corresponding maximum value is $\sqrt{6} + \sqrt{1.5}$. From this, the maximum winning probability in quantum scenario can be written as,

$$\begin{aligned} & \frac{1}{4} \left[2 + \frac{1}{4} + \frac{1}{4} (\|u_0\| \|2v_0 - v_1\| + \|u_1\| \|v_0 + v_1\|) \right] \\ & \leq \frac{1}{4} \left[\frac{9}{4} + \frac{1}{4} \times (\sqrt{6} + \sqrt{1.5}) \right] \\ & \approx 0.792 \end{aligned}$$

This implies that one can find some values of $\theta_0, \theta_1, \psi_0$ and ψ_1 for which the game can be won with a probability of 0.792 in the quantum scenario. Similarly one can show the same upper bound for some other games of this group for which the maximum classical success probability is 0.75. Therefore the maximum separation for this class of games is $(0.792 - 0.75) \approx 0.042$.

From this discussion, it is clear that for all the games corresponding to partitions $4 + 3 + 1 + 1$, $4 + 2 + 2 + 1$ and $3 + 2 + 2 + 2$, there are no chances of getting quantum advantage. But for the partition $3 + 3 + 2 + 1$, there are some games which provide quantum advantage with a separation around 0.042.

7.8.3 Games Corresponding to 10 Successful Outcomes

Proceeding in similar way as the analysis of the 8 successful outcomes, the maximum classical and quantum success probabilities that one can achieve for each of the partitions of the 10 successful outcomes are mentioned in Table 7.5. From this result, one can easily check that the quantum advantage can be achieved only for some of the games corresponding to partition $3 + 3 + 3 + 1$ with a separation around 0.05. For simplicity, here we consider one of these games having quantum advantage and analyze its performance in both classical and quantum scenarios.

Partitions	Max. classical success prob.	Max. quantum success prob.	Corresponding Separation
4+4+1+1	1.0	1.0	NA
4+2+2+2	1.0	1.0	NA
3+3+2+2	1.0	1.0	NA
4+3+2+1	1.0	1.0	NA
3+3+3+1	0.75	0.8	0.05
	1.0	1.0	NA

Table 7.5: Analysis of partitions for 10 successful outcomes

A game for partition $3 + 3 + 3 + 1$ having quantum advantage

From the results of table 7.5, it is clear that for the games having 10 successful outcomes, quantum advantage can be achieved only for some of the games having partition $3 + 3 + 3 + 1$. Here we consider the following game which can't be won with certainty in classical scenario.

Input	Corresponding output
00	00
01	00, 10, 11
10	00, 01, 11
11	01, 10, 11

From the strategies mentioned in subsection 7.5, one can easily check that the maximum classical success probability for this game is 0.75 and one of the strategies to get this success probability is $a = 0$ and $b = 0$.

Similarly from the discussion of subsection 7.6, one can easily check that the expression for quantum success probability of this mentioned game is of the form

$$\begin{aligned} & \frac{1}{4} \left[\frac{1}{2} \cos^2 \alpha + \frac{1}{2} + \frac{1}{2} \cos^2 \beta + \frac{1}{2} + \frac{1}{2} \cos^2 \gamma + \frac{1}{2} + \frac{1}{2} \sin^2 \delta \right] \\ &= \frac{1}{4} \left[\frac{5}{2} + \frac{1}{4} (\cos 2\alpha + \cos 2\beta + \cos 2\gamma - \cos 2\delta) \right] \end{aligned}$$

where $\alpha = (\theta_0 - \psi_0)$, $\beta = (\theta_0 - \psi_1)$, $\gamma = (\theta_1 - \psi_0)$ and $\delta = (\theta_1 - \psi_1)$.

One can easily think of the cosines as the inner products between unit vectors. In that case, one can rewrite the above expression as

$$\begin{aligned} & \frac{1}{4} \left[\frac{5}{2} + \frac{1}{4} (u_0 v_0 + u_0 v_1 + u_1 v_0 - u_1 v_1) \right] \\ & \leq \frac{1}{4} \left[\frac{5}{2} + \frac{1}{4} (\|u_0\| \|v_0 + v_1\| + \|u_1\| \|v_0 - v_1\|) \right] \end{aligned}$$

Now, $\|u_0\| \|v_0 + v_1\| + \|u_1\| \|v_0 - v_1\| \leq \|v_0 + v_1\| + \|v_0 - v_1\| \leq 2\sqrt{2}$.

Hence the winning probability in quantum scenario $\leq \frac{1}{4} \left[\frac{5}{2} + \frac{1}{4} \times 2\sqrt{2} \right] \approx 0.80$.

This implies that one can find some values of $\theta_0, \theta_1, \psi_0$ and ψ_1 for which the game can be won with a probability of 0.80 in the quantum scenario. Similarly one can show the same upper bound for some other games of this group for which the maximum classical success probability is 0.75. Therefore the maximum separation for this class of games is $(0.80 - 0.75) \approx 0.05$.

From this discussion, it is clear that for all the games corresponding to partitions $4 + 4 + 1 + 1$, $4 + 2 + 2 + 2$, $3 + 2 + 2 + 2$ and $4 + 3 + 2 + 1$, there are no chances of getting quantum advantage. But for the partition $3 + 3 + 3 + 1$, there are some games which provide quantum advantage with a separation around 0.05.

7.8.4 Games Corresponding to 11 Successful Outcomes

Proceeding in similar way as the analysis of the 8 successful outcomes, the maximum classical and quantum success probabilities that one can achieve for each of the partitions of the 11 successful outcomes are mentioned in the Table 7.6. From this result, one can easily check that there are no games corresponding to 11 successful outcomes for which quantum advantage can be achieved.

So for all the games corresponding to 11 successful outcomes, there are no chances

Partitions	Max. classical success prob.	Max. quantum success prob.	Corresponding Separation
4+4+2+1	1.0	1.0	NA
4+3+3+1	1.0	1.0	NA
4+3+2+2	1.0	1.0	NA
3+3+3+2	1.0	1.0	NA

Table 7.6: Analysis of partitions for 11 successful outcomes

of getting any advantage in quantum success probability as compared to the classical one.

7.8.5 Games Corresponding to 12 or More Successful Outcomes

One can easily verify that each of the partitions for 12 successful outcomes is an extension of some partitions corresponding to 11 successful outcomes. As all the games corresponding to 11 successful outcomes can be won classically with certainty, there is no chance of getting quantum advantage for any of the games having 12 successful outcomes. Similarly one can also argue the same statement for 13 or more successful outcomes.

For this reason, the games having 12 or more successful outcomes can't achieve quantum advantage.

7.8.6 Games Corresponding to 7 Successful Outcomes

Proceeding in similar way as the analysis of the 8 successful outcomes, the maximum classical and quantum success probabilities that one can achieve for each of the partitions of the 7 successful outcomes are mentioned in the Table 7.7. From these results, one can easily check that quantum advantage can be achieved only for some of the games corresponding to partition $2 + 2 + 2 + 1$ with a separation around 0.012. For simplicity, here we consider one of these games having quantum advantage and analyze its performance.

Partitions	Max. classical success prob.	Max. quantum success prob.	Corresponding Separation
3+2+1+1	0.75	0.75	NA
	1.0	1.0	NA
2+2+2+1	0.75	0.762	0.012
	1.0	1.0	NA

Table 7.7: Analysis of partitions for 7 successful outcomes

A game for partition 2 + 2 + 2 + 1 having quantum advantage

From the results of table 7.7, it is clear that for the games having 7 successful outcomes, quantum advantage can be achieved only for some of the games having partition 2 + 2 + 2 + 1. Here we consider the following game which can't be won with certainty in the classical scenario. From the strategies mentioned in subsection 7.5,

Input	Corresponding output
00	00, 11
01	01, 10
10	11
11	00, 11

one can easily check that the maximum classical success probability for this game is 0.75 and one of the strategies to get this success probability is $a = 1$ and $b = 1$.

Similarly from the discussion of subsection 7.6, one can easily check that the expression for quantum success probability of this mentioned game is of the form

$$\begin{aligned} & \frac{1}{4} \left[\cos^2 \alpha + \sin^2 \beta + \frac{1}{2} \cos^2 \gamma + \cos^2 \delta \right] \\ &= \frac{1}{4} \left[\frac{7}{4} + \frac{1}{4} (2 \cos 2\alpha - 2 \cos 2\beta + \cos 2\gamma + 2 \cos 2\delta) \right] \end{aligned}$$

where $\alpha = (\theta_0 - \psi_0)$, $\beta = (\theta_0 - \psi_1)$, $\gamma = (\theta_1 - \psi_0)$ and $\delta = (\theta_1 - \psi_1)$.

One can easily think of the cosines as the inner products between unit vectors. In that case, one can rewrite the above expression as

$$\begin{aligned} & \frac{1}{4} \left[\frac{7}{4} + \frac{1}{4} (2u_0v_0 - 2u_0v_1 + u_1v_0 + 2u_1v_1) \right] \\ & \leq \frac{1}{4} \left[\frac{7}{4} + \frac{1}{4} (\|u_0\| \|2v_0 - 2v_1\| + \|u_1\| \|v_0 + 2v_1\|) \right] \end{aligned}$$

Let us assume that $\langle v_0, v_1 \rangle = (a + ib)$ and $\langle v_1, v_0 \rangle = (a - ib)$. Then one can easily check that $\|2v_0 - 2v_1\| = 2\|v_0 - v_1\| = 2\sqrt{2 - 2a}$ and $\|v_0 + 2v_1\| = \sqrt{5 + 4a}$. From these expressions, one can easily calculate that the expression $2\sqrt{2 - 2a} + \sqrt{5 + 4a}$ attains maximum value for $a = -\frac{1}{2}$ and the corresponding maximum value is $3\sqrt{3}$. From this, the maximum winning probability in quantum scenario can be written as,

$$\begin{aligned} & \frac{1}{4} \left[\frac{7}{4} + \frac{1}{4} (\|u_0\| \|2v_0 - 2v_1\| + \|u_1\| \|v_0 + 2v_1\|) \right] \\ & \leq \frac{1}{4} \left[\frac{7}{4} + \frac{1}{4} \times 3\sqrt{3} \right] \\ & \approx 0.762 \end{aligned}$$

This implies that one can find some values of $\theta_0, \theta_1, \psi_0$ and ψ_1 for which the game can be won with a probability of 0.762 in the quantum scenario. Similarly one can explore that the same upper bound can be achieved for all the other games of this group for which quantum advantage can be achieved and the maximum classical success probability is 0.75. Therefore the maximum separation for this class of games is $(0.762 - 0.75) \approx 0.012$.

From this discussion, it is clear that for all the games corresponding to partition $3+2+1+1$, there are no chances of getting quantum advantage. But for the partition $2+2+2+1$, there are some games which provide quantum advantage with a separation around 0.012.

7.8.7 Games Corresponding to 6 Successful Outcomes

Proceeding in similar way as the analysis of the 8 successful outcomes, the maximum classical and quantum success probabilities that one can achieve for each of the partitions of the 6 successful outcomes are mentioned in the Table 7.8. From these results, one can easily check that quantum advantage can be achieved only for some of the games corresponding to partition $3+1+1+1$ with a separation around 0.05. Here we consider one of these games having quantum advantage and analyze its performance in both classical and quantum scenarios.

Partitions	Max. classical success prob.	Max. quantum success prob.	Corresponding Separation
3+1+1+1	0.5	0.55	0.05
	0.75	0.75	NA
	1.0	1.0	NA
2+2+1+1	0.75	0.75	NA
	1.0	1.0	NA

Table 7.8: Analysis of partitions for 6 successful outcomes

A game for partition 3 + 1 + 1 + 1 having quantum advantage

From the results of table 7.8, it is clear that for the games having 6 successful outcomes, quantum advantage can be achieved only for some of the games having partition 3 + 1 + 1 + 1. Here we consider the following game which can't be won with certainty in the classical scenario.

Input	Corresponding output
00	00, 01, 10
01	11
10	01
11	10

From the strategies mentioned in subsection 7.5, one can easily check that the maximum classical success probability for this game is 0.5 and one of the strategies to get this success probability is $a = 0$ and $b = 1$.

Similarly from the discussion of subsection 7.6, one can easily check that the expression for quantum success probability of this mentioned game is of the form

$$\begin{aligned} & \frac{1}{4} \left[\frac{1}{2} + \frac{1}{2} \sin^2 \alpha + \frac{1}{2} \cos^2 \beta + \frac{1}{2} \sin^2 \gamma + \frac{1}{2} \sin^2 \delta \right] \\ &= \frac{1}{4} \left[\frac{3}{2} + \frac{1}{4} (-\cos 2\alpha + \cos 2\beta - \cos 2\gamma - \cos 2\delta) \right] \end{aligned}$$

where $\alpha = (\theta_0 - \psi_0)$, $\beta = (\theta_0 - \psi_1)$, $\gamma = (\theta_1 - \psi_0)$ and $\delta = (\theta_1 - \psi_1)$.

One can easily think of the cosines as the inner products between two unit vectors. In that case, one can rewrite the above expression as

$$\frac{1}{4} \left[\frac{3}{2} + \frac{1}{4} (-u_0 v_0 + u_0 v_1 - u_1 v_0 - u_1 v_1) \right]$$

$$\leq \frac{1}{4} \left[\frac{3}{2} + \frac{1}{4} (\|u_0\| \|v_0 - v_1\| + \|u_1\| \|v_0 + v_1\|) \right]$$

Now, $\|u_0\| \|v_0 - v_1\| + \|u_1\| \|v_0 + v_1\| \leq \|v_0 - v_1\| + \|v_0 + v_1\| \leq 2\sqrt{2}$.

Hence the winning probability $\leq \frac{1}{4} \left[\frac{3}{2} + \frac{1}{4} \times 2\sqrt{2} \right] \approx 0.55$.

This implies that one can find some values of $\theta_0, \theta_1, \psi_0$ and ψ_1 for which the game can be won with a probability of 0.55 in the quantum scenario. Similarly one can explore that the same upper bound can be achieved for all the other games of this group for which quantum advantage can be achieved and the maximum classical success probability is 0.5. Therefore the maximum separation for this class of games is $(0.55 - 0.5) \approx 0.05$.

From this discussion, it is clear that for all the games corresponding to partition $2+2+1+1$, there are no chances of getting quantum advantage. But for the partition $3+1+1+1$, there are some games which provide quantum advantage with a separation around 0.05.

7.8.8 Games Corresponding to 5 Successful Outcomes

Proceeding in similar way as the analysis of the 8 successful outcomes, the maximum classical and quantum success probabilities that one can achieve for each of the partitions of the 5 successful outcomes are mentioned in the Table 7.9. From these results, one can easily check that quantum advantage can be achieved for the games corresponding to partition $2 + 1 + 1 + 1$ with a separation around 0.042. Here we consider one of these games having quantum advantage and analyze its performance in both classical and quantum scenarios.

Partitions	Max. classical success prob.	Max. quantum success prob.	Corresponding Separation
2+1+1+1	0.5	0.542	0.042
	0.75	0.75	NA
	1.0	1.0	NA

Table 7.9: Analysis of partitions for 5 successful outcomes

A game for partition 2 + 1 + 1 + 1 having quantum advantage

From the results of table 7.9, it is clear that for the games having 5 successful outcomes, quantum advantage can be achieved only for some of the games having partition 2 + 1 + 1 + 1. Here we consider the following game which can't be won with certainty in the classical scenario.

Input	Corresponding output
00	01, 10
01	11
10	01
11	10

From the strategies mentioned in subsection 7.5, one can easily check that the maximum classical success probability for this game is 0.5 and one of the strategies to get this success probability is $a = 0$ and $b = 1$.

Similarly from the discussion of subsection 7.6, one can easily check that the expression for quantum success probability of this mentioned game is of the form

$$\begin{aligned} & \frac{1}{4} \left[\sin^2 \alpha + \frac{1}{2} \cos^2 \beta + \frac{1}{2} \sin^2 \gamma + \frac{1}{2} \sin^2 \delta \right] \\ &= \frac{1}{4} \left[\frac{5}{4} + \frac{1}{4} (-2 \cos 2\alpha + \cos 2\beta - \cos 2\gamma - \cos 2\delta) \right] \end{aligned}$$

where $\alpha = (\theta_0 - \psi_0)$, $\beta = (\theta_0 - \psi_1)$, $\gamma = (\theta_1 - \psi_0)$ and $\delta = (\theta_1 - \psi_1)$.

As one can think of the cosines as the inner products between unit vectors, the above expression can be rewritten as,

$$\begin{aligned} & \frac{1}{4} \left[\frac{5}{4} + \frac{1}{4} (-2u_0v_0 + u_0v_1 - u_1v_0 - u_1v_1) \right] \\ & \leq \frac{1}{4} \left[\frac{5}{4} + \frac{1}{4} (\| -2v_0 + v_1 \| + \| v_0 + v_1 \|) \right] \end{aligned}$$

Let us assume that $\langle v_0, v_1 \rangle = (a + ib)$ and $\langle v_1, v_0 \rangle = (a - ib)$. Then one can easily check that $\| -2v_0 + v_1 \| = \sqrt{5 - 4a}$ and $\| v_0 + v_1 \| = \sqrt{2 + 2a}$. From these expressions, one can easily calculate that the expression $\sqrt{2 + 2a} + \sqrt{5 - 4a}$ attains maximum value for $a = -\frac{1}{4}$ and the corresponding maximum value is $\sqrt{6} + \sqrt{1.5}$.

From this, the maximum winning probability in quantum scenario can be written as,

$$\begin{aligned} & \frac{1}{4} \left[\frac{5}{4} + \frac{1}{4} (\| -2v_0 + v_1 \| + \| v_0 + v_1 \|) \right] \\ & \leq \frac{1}{4} \left[\frac{5}{4} + \frac{1}{4} \times (\sqrt{6} + \sqrt{1.5}) \right] \\ & \approx 0.542 \end{aligned}$$

This implies that one can find some values of $\theta_0, \theta_1, \psi_0$ and ψ_1 for which the game can be won with a probability of 0.542 in the quantum scenario. Similarly one can explore that the same upper bound can be achieved for all the other games of this group for which quantum advantage can be achieved and the maximum classical success probability is 0.5. Therefore the maximum separation for this class of games is $(0.542 - 0.5) \approx 0.042$. From this discussion, it is clear that for all the games corresponding to partition $2+1+1+1$, there are some games which provide quantum advantage with a separation around 0.042.

7.8.9 Games Corresponding to 4 Successful Outcomes

Proceeding in similar way as the analysis of the 8 successful outcomes, the maximum classical and quantum success probabilities that one can achieve for the partition of the 4 successful outcomes are mentioned in the Table 7.10. From these results, one can easily check that there are no games corresponding to 4 successful outcomes for which quantum advantage can be achieved.

Partitions	Max. classical success prob.	Max. quantum success prob.	Corresponding Separation
1+1+1+1	0.5	0.5	NA
	0.75	0.75	NA
	1.0	1.0	NA

Table 7.10: Analysis of partition for 4 successful outcomes

So for all the games corresponding to 4 successful outcomes, there are no chances of getting any advantage in quantum success probability as compared to the classical one.

7.8.10 Games Corresponding to 3 or Less Successful Outcomes

One cannot divide 3 or less number of successful outcomes into four parts such that each part has atleast one outcome. Hence for these class of games, one can easily argue from the discussion in subsection 7.6 that the maximum quantum success probability is always less than 0.5 and there is no chance of getting quantum advantage.

No. of succ. outcome	Partition with quant. advantage	A game corr. to the partition (in ANF form)	Max. classical succ. prob.	Corr. classical strategy	Max. quant. succ. prob.	Corr. quantum strategy	Max. Separation
10	3+3+3+1	$a \oplus b \oplus xy \oplus xb \oplus ya \oplus ab \oplus xya \oplus xyb$	0.75	a=0, b=0	0.80	$\theta_0 = 0, \theta_1 = \frac{\pi}{4}$ $\psi_0 = \frac{\pi}{8}, \psi_1 = \frac{7\pi}{8}$	0.05
9	3+3+2+1	$x \oplus a \oplus b \oplus xy \oplus xa \oplus xb \oplus ya \oplus yb \oplus xyb \oplus xab \oplus yab \oplus xyab$	0.75	a=0, b=0	0.792	$\theta_0 = 0, \theta_1 = \frac{21\pi}{100}$ $\psi_0 = \frac{\pi}{8}, \psi_1 = \frac{7\pi}{20}$	0.042
8	2+2+2+2	$a \oplus b \oplus xy$	0.75	a=0, b=0	0.853	$\theta_0 = 0, \theta_1 = \frac{\pi}{4}$ $\psi_0 = \frac{\pi}{8}, \psi_1 = \frac{7\pi}{8}$	0.103
8	3+2+2+1	$x \oplus a \oplus xy \oplus xa \oplus xb \oplus yb \oplus ab \oplus xya \oplus xyb \oplus yab$	0.75	a=0, b=0	0.78	$\theta_0 = 0, \theta_1 = \frac{7\pi}{50}$ $\psi_0 = \frac{5\pi}{6}, \psi_1 = \frac{7\pi}{100}$	0.03
7	2+2+2+1	$x \oplus y \oplus a \oplus b \oplus xa \oplus xb \oplus xya \oplus xyb \oplus xab \oplus xyab$	0.75	a=1, b=1	0.762	$\theta_0 = 0, \theta_1 = \frac{\pi}{3}$ $\psi_0 = \frac{2\pi}{25}, \psi_1 = \frac{21\pi}{50}$	0.012
6	3+1+1+1	$x \oplus y \oplus xy \oplus xb \oplus ab \oplus xya \oplus xyb$	0.5	a=0, b=1	0.55	$\theta_0 = 0, \theta_1 = \frac{\pi}{4}$ $\psi_0 = \frac{5\pi}{8}, \psi_1 = \frac{7\pi}{8}$	0.05
5	2+1+1+1	$1 \oplus a \oplus b \oplus xa \oplus ya \oplus yb \oplus xab \oplus yab \oplus xyab$	0.5	a=0, b=1	0.542	$\theta_0 = 0, \theta_1 = \frac{21\pi}{100}$ $\psi_0 = \frac{14\pi}{25}, \psi_1 = \frac{17\pi}{20}$	0.042

Table 7.11: List of partitions and the corresponding nonlocal games (in ANF form) which offer quantum advantage.

7.9 Conclusion

In this chapter, we explore the performance of all possible binary input binary output two-party nonlocal games in terms of partitions of the total number of successful outcomes to check whether there exist any such games which offer a quantum advantage

with maximum quantum success probability greater than 0.85. The maximum classical and the maximum quantum success probabilities for the games corresponding to each of those partitions are mentioned in Table 7.11. From our analysis, we found that there are some binary input binary output two-party nonlocal games (other than the CHSH game) that offer quantum advantage but the CHSH game has the maximum quantum success probability (also with a maximum separation of around 0.1) among all these games. Moreover, we also add the characterization of classical strategies to any n -party nonlocal game. However, this chapter does not demonstrate any quantum advantage for an n -party nonlocal game.

Contents

8.1 Brief Summary & Further Research Directions	185
--	------------

In this concluding chapter, we summarize the preceding ones and draw relevant conclusions from various aspects explored throughout this thesis. The main emphasis of this thesis revolves around subjects pertaining to Boolean functions. In this section, we emphasize our significant contributions, enhancements, and expansions to current methodologies.

8.1 Brief Summary & Further Research Directions

This thesis enriches this comprehension through five thorough chapters, which we will summarize below.

- In Chapter 3 we have explored specific constraints associated with Arbiter PUFs. Our findings reveal that by exhaustively adjusting the delay parameters, an n -length Arbiter PUFs can only produce a negligible portion of Boolean functions. We also make a combinatorial analysis of Arbiter PUFs and present several existence and non-existence results in this direction. Then we explore efficient techniques towards distinguishing whether a Boolean function truth table can indeed be fabricated through an Arbiter PUF or not. Our efficient algorithms clearly show that the Boolean functions generated from n -length Arbiter PUFs are not well sampled from the generic Boolean function class.

Further we have looked at autocorrelation in certain restricted sense and presented relevant results in this direction. It is known that the autocorrelation property of Boolean functions generated out of Arbiter PUFs is quite biased in certain cases. Interestingly, here we note that under certain constraints on the weights of inputs, along with the difference in a specific input bit, such biases vanish. That is, such a simple model of Arbiter PUFs provide good cryptographic parameters in terms of differential analysis if certain restrictions on the input challenge pairs are imposed.

- In Chapter 4 we have studied the number of Boolean functions class generated from the XOR of two PUFs or the XOR of two threshold functions. Among the two variants of XOR-PUFs, it is shown that the XOR of two n -length Arbiter PUFs with the same input can generate a significant number of Boolean functions. However, the other variant can only generate a negligible number of Boolean functions. In the final part of this chapter, we provide the theoretical probability of equality of the outputs from the PUF with the combiner function corresponding to two different inputs. This result motivates us to look into the class of Boolean functions generated from a PUF via a combiner function. Our analysis of XOR-PUF presents the following challenges for readers to consider:

1. XOR of n number of Arbiter PUFs of length n can generate all m -variable Boolean functions such that $m \leq n$ i.e., $\mathcal{B}_m^{\text{n-XOR-PUF}} = \mathcal{B}_m$, where $m \leq n$.
2. $|\mathcal{B}_{n+1}^{\text{n-XOR-PUF}}| = 2^{2^{n+1}} - 2$.

- In Chapter 5, we have performed analysis on PA-PUF. At first we have developed the mathematical model of PA-PUF. From the mathematical model of PA-PUF we started analysing the cryptographic properties of PA-PUF. We have shown that the set of Boolean functions generated from PUF is a subset of the set Boolean functions generated from PA-PUF for any length. We have also proposed an algorithm to determine the set of Boolean functions which is generated from PA-PUF with length n . Further, we have studied the bias in the output bit of PA-PUF when certain challenge input bit is flipped. This chapter focuses on the estimation of the probabilities for two cases (1) when the sign of last bit is altered, (2) when the sign of second last bit is altered. We have shown that the randomness in the output bit of PA-PUF is improved

over the randomness in the output bit of classical PUF. In the final part of this chapter, we conducted an in-depth analysis of the cryptographic properties of PA-PUF. Our findings indicate that PA-PUF possesses superior cryptographic characteristics compared to other existing PUFs, demonstrating greater robustness against modeling attacks. Our analysis on PA-PUF keeps the following challenges open for the readers:

1. What is the optimal number of paths required in the circuit of PA-PUF for generating the entire set \mathcal{B}_n for every n ?
 2. Theoretically estimate the $Pr[z_C = z_{\tilde{C}}]$ when the sign of any arbitrary $c[i]$ is altered, $0 \leq i \leq n - 3$.
- Then we move towards different applications. In Chapter 6, we have presented several critical points related to *BoolTest* [SKŠ17], which is considered to be a method to evaluate the randomness of a stream of data. In this connection, we present combinatorial results related to identifying the most suitable Boolean functions in maximizing the Z -score that could not be achieved in the heuristic presented for *BoolTest* [SKŠ17]. Our Algorithm 6 finds the best Boolean function having the maximum Z -Score in $O(N \log N)$ time, given N amount of data. While we solve certain combinatorial problems related to *BoolTest*, the caveat is, this test is not sufficient to conclude on randomness or non-randomness of a given stream of data. Certain statistical interpretations have been discussed in [SKŠ17, Section 5], but we believe that this tool needs further evaluation. For example, one may consider cross-testing based on the generated polynomials from *BoolTest* [SKŠ17] or by our method that we have discussed in Section 6.4.3. Further analysis in this regard might provide a better understanding of this domain, which we put forward for future research.
 - Considering another application, in Chapter 7, we study certain aspects of non-local games. In particular, identified seven partitions (over all possible games having at least one successful outcome for each possible input) such that the games corresponding to those partitions offer a quantum advantage. The maximum classical and the maximum quantum success probabilities for the games corresponding to each of those partitions are mentioned in Table 7.11. We also mention an example of such a game (in Algebraic Normal Form) for each of

those partitions. It is well known that the CHSH game is used to certify untrusted devices in the device-independent scenario. It is also known that the required sample size for device-independent testing is inversely proportional to the success probability of the corresponding nonlocal game. Although the maximum success probability for the CHSH game using quantum resources is less than 1 (around 0.85), so far no other two-party nonlocal game is used for device-independent testing. To the best of our knowledge, it was also unknown whether there exists any other binary input binary output two-party nonlocal game which offers a quantum advantage. To answer all these questions, in this article, we explore the performance of all possible binary input binary output two-party nonlocal games in terms of partitions of the total number of successful outcomes to check whether there exist any such games which offer a quantum advantage with maximum quantum success probability greater than 0.85. From our analysis, we found that there are some binary input binary output two-party nonlocal games (other than the CHSH game) that offer quantum advantage but the CHSH game has the maximum quantum success probability (also with a maximum separation of around 0.1) among all these games. Further study for three (or more) party nonlocal games will be an interesting research work in this direction. We additionally include the characterization of classical strategies for any n -party nonlocal game.

However, it is important to note that this chapter does not illustrate any quantum advantage in the context of n -party nonlocal games and that may be considered in future efforts.

In summary, this thesis wants to make the study of Boolean functions better by giving us more understanding, new viewpoints, and pointing out things we still don't know. We hope that our research will inspire more studies and help make small steps forward and big discoveries in this interesting area of research.

Bibliography

- [AES01] Specification for the Advanced Encryption Standard (AES). Federal Information Processing Standards Publication 197, 2001. doi: <http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf>
- [AHC22] N. N. Anandakumar, M. S. Hashmi, and M. A. Chaudhary. Implementation of Efficient XOR Arbiter PUF on FPGA With Enhanced Uniqueness and Security. *IEEE Access*, 10:129832–129842, 2022. doi: <https://doi.org/10.1109/ACCESS.2022.3228635>
- [AHS20] N. N. Anandakumar, M. S. Hashmi, and S. K. Sanadhya. Efficient and Lightweight FPGA-based Hybrid PUFs with Improved Performance. *Microprocess. Microsystems*, 77:103180, 2020. doi: <https://doi.org/10.1016/j.micpro.2020.103180>
- [AHS17] N. N. Anandakumar, M. S. Hashmi, and S. K. Sanadhya. Compact Implementations of FPGA-based PUFs with Enhanced Performance. In *30th International Conference on VLSI Design and 16th International Conference on Embedded Systems, VLSID 2017, Hyderabad, India, January 7-11, 2017*, pages 161–166. IEEE Computer Society, 2017. doi: <https://doi.org/10.1109/VLSID.2017.7>
- [AHS22] N. N. Anandakumar, M. S. Hashmi, and S. K. Sanadhya. Design and Analysis of FPGA-based PUFs with Enhanced Performance for

- Hardware-oriented Security. *J. Emerg. Technol. Comput. Syst.*, 18(4), oct 2022. doi: <https://doi.org/10.1145/3517813>
- [AHT21] N. N. Anandakumar, M. S. Hashmi, and M Tehranipoor. FPGA-based Physical Unclonable Functions: A comprehensive overview of theory and architectures. *Integration*, 81:175–194, 2021. doi: <https://doi.org/10.1016/j.vlsi.2021.06.001>
- [BB14] C. H. Bennett and G. Brassard. Quantum cryptography: Public key distribution and coin tossing. *Theoretical Computer Science*, 560:7–11, 2014. Theoretical Aspects of Quantum Cryptography celebrating 30 years of BB84. doi: <https://doi.org/10.1016/j.tcs.2014.05.025>
- [BBT05] G. Brassard, A. Broadbent, and A. Tapp. Quantum Pseudo-Telepathy. *Foundations of Physics*, 35(11):18771907, November 2005. doi: <http://dx.doi.org/10.1007/s10701-005-7353-4>
- [BEB] R. G. Brown, D. Eddelbuettel, and D. Bauer. *Dieharder: A Random Number Test Suite (Version 3.31.1)*. 2014. doi: <https://webhome.phy.duke.edu/~rgb/General/dieharder.php>
- [Bec15] G. T. Becker. The Gap Between Promise and Reality: On the Insecurity of XOR Arbiter PUFs. In Tim Güneysu and Helena Handschuh, editors, *Cryptographic Hardware and Embedded Systems - CHES 2015 - 17th International Workshop, Saint-Malo, France, September 13-16, 2015, Proceedings*, volume 9293 of *Lecture Notes in Computer Science*, pages 535–555. Springer, 2015. doi: https://doi.org/10.1007/978-3-662-48324-4_27
- [Bel64] J. S. Bell. On the Einstein Podolsky Rosen paradox. *Physique Physique Fizika*, 1:195–200, Nov 1964. doi: <https://link.aps.org/doi/10.1103/PhysicsPhysiqueFizika.1.195>
- [BFSK11] C. Brzuska, M. Fischlin, H. Schröder, and S. Katzenbeisser. Physically Uncloneable Functions in the Universal Composition Framework. In Phillip Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*,

pages 51–70, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. doi: https://doi.org/10.1007/978-3-642-22792-9_4

- [BM18] J. Basak and S. Maitra. Clauser - Horne - Shimony - Holt versus three-party pseudo-telepathy: on the optimal number of samples in device-independent quantum private query. *Quantum Information Processing*, 17, 2018. doi: <https://doi.org/10.1007/s11128-018-1849-2>
- [BRS⁺10] L. Bassham, A. Rukhin, J. Soto, J. Nechvatal, M. Smid, S. Leigh, M. Levenson, M. Vangel, N. Heckert, and D. Banks. A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications, 2010-09-16 2010. https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=906762
- [CaR17] C. Carlet, P. Méaux, and Y. Rotella. Boolean functions with restricted input and their robustness; application to the FLIP cipher. *IACR Transactions on Symmetric Cryptology*, 09 2017. doi: <https://doi.org/10.13154/tosc.v2017.i3.192-227>
- [CCF⁺18] A. Canteaut, S. Carпов, C. Fontaine, T. Lepoint, M. Naya-Plasencia, P. Paillier, and R. Sirdey. Stream Ciphers: A Practical Solution for Efficient Homomorphic-Ciphertext Compression. *Journal of Cryptology*, 31, 01 2018. doi: <https://doi.org/10.1007/s00145-017-9273-9>
- [CCKM16] U. Chatterjee, R. S. Chakraborty, H. Kapoor, and D. Mukhopadhyay. Theory and Application of Delay Constraints in Arbiter PUF. *ACM Trans. Embed. Comput. Syst.*, 15(1):10:1–10:20, 2016. doi: <https://doi.org/10.1145/2815621>
- [CCM17] U. Chatterjee, R. Chakraborty, and D. Mukhopadhyay. A PUF-based secure communication protocol for IoT. *ACM Transactions on Embedded Computing Systems*, 16:1–25, 04 2017. doi: <https://doi.org/10.1145/3005715>
- [CCMC20] U. Chatterjee, S. Chatterjee, D. Mukhopadhyay, and R. S. Chakraborty. Machine Learning Assisted PUF Calibration for Trustworthy Proof of

Sensor Data in IoT. *ACM Trans. Des. Autom. Electron. Syst.*, 25(4), jun 2020. doi: <https://doi.org/10.1145/3393628>

- [CCMH21] D. Chatterjee, U. Chatterjee, D. Mukhopadhyay, and A. Hazra. SA-CReD: An Attack Framework on SAC Resistant Delay-PUFs leveraging Bias and Reliability Factors. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*, pages 85–90, 2021. doi: <https://doi.org/10.1109/DAC18074.2021.9586249>
- [CGS⁺19] U. Chatterjee, V. Govindan, R. Sadhukhan, D. Mukhopadhyay, R. S. Chakraborty, D. Mahata, and M. Prabhu. Building PUF Based Authentication and Key Exchange Protocol for IoT Without Explicit CRPs in Verifier Database. *IEEE Trans. Dependable Secur. Comput.*, 16(3):424–437, 2019. doi: <https://doi.org/10.1109/TDSC.2018.2832201>
- [CHSH69] J. F. Clauser, A. Horne, A. Shimony, and R. A. Holt. Proposed Experiment to Test Local Hidden-Variable Theories. *Phys. Rev. Lett.*, 23:880–884, Oct 1969. doi: <https://link.aps.org/doi/10.1103/PhysRevLett.23.880>
- [Cir80] B. S. Cirel’son. Quantum generalizations of Bell’s inequality. *Letters in Mathematical Physics*, 4(2):93–100, Mar 1980. doi: <https://doi.org/10.1007/BF00417500>
- [CM14] R. Cleve and R. Mittal. Characterization of Binary Constraint System Games. In Javier Esparza, Pierre Fraigniaud, Thore Husfeldt, and Elias Koutsoupias, editors, *Automata, Languages, and Programming*, pages 320–331, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg. doi: https://doi.org/10.1007/978-3-662-43948-7_27
- [Dev09] S. Devadas. Physical Unclonable Functions and Secure Processors. In Christophe Clavier and Kris Gaj, editors, *Cryptographic Hardware and Embedded Systems - CHES 2009*, pages 65–65, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg. doi: https://doi.org/10.1007/978-3-642-04138-9_5

- [DPM19] N. Das, G. Paul, and A. Maitra. Dimensionality distinguishers. *Quant. Inf. Process.*, 18(171), 2019. doi: <https://doi.org/10.1007/s11128-019-2279-5>
- [DR02] J. Daemen and V. Rijmen. *The design of Rijndael: AES — the Advanced Encryption Standard*. Springer-Verlag, 2002. doi: <https://doi.org/10.1007/978-3-662-04722-4>
- [DV13] J. Delvaux and I. Verbauwhede. Side channel modeling attacks on 65nm arbiter PUFs exploiting CMOS device noise. In *2013 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, pages 137–142, 2013. doi: <https://doi.org/10.1109/HST.2013.6581579>
- [EHFCS21] M. El-Hajj, A. Fadlallah, M. Chamoun, and A. Serhrouchni. A taxonomy of PUF Schemes with a novel Arbiter-based PUF resisting machine learning attacks. *Computer Networks*, 194:108133, 2021. doi: <https://doi.org/10.1016/j.comnet.2021.108133>
- [Gas03] B. Gassend. Physical random functions. 2003. <https://api.semanticscholar.org/CorpusID:111025374>
- [GCDD02] B. Gassend, D. Clarke, M. V. Dijk, and S. Devadas. Silicon physical random functions. In *Conference on Computer and Communications Security*, 2002. doi: <https://doi.org/10.1145/586110.586132>
- [GCL⁺21] C. Gu, C. H. Chang, W. Liu, S. Yu, Y. Wang, and M. O'Neill. A Modeling Attack Resistant Deception Technique for Securing Lightweight-PUF-Based Authentication. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 40(6):1183–1196, 2021. doi: <https://doi.org/10.1109/TCAD.2020.3036807>
- [GDNC94] H. Gustafson, E. Dawson, L. Nielsen, and W. Caelli. A computer package for measuring the strength of encryption algorithms. *Computers & Security*, 13(8):687–697, 1994. doi: [https://doi.org/10.1016/0167-4048\(94\)90051-5](https://doi.org/10.1016/0167-4048(94)90051-5)

- [GHO17] C. Gu, N. Hanley, and M. O’neill. Improved Reliability of FPGA-Based PUF Identification Generator Design. *ACM Trans. Reconfigurable Technol. Syst.*, 10(3), may 2017. doi: <https://doi.org/10.1145/3053681>
- [GHZ89] D. M. Greenberger, M. A. Horne, and A. Zeilinger. *Going Beyond Bell’s Theorem*, pages 69–72. Springer Netherlands, Dordrecht, 1989. doi: https://doi.org/10.1007/978-94-017-0849-4_10
- [GKST07] J. Guajardo, S.S. Kumar, G.J. Schrijen, and P. Tuyls. FPGA Intrinsic PUFs and Their Use for IP Protection. In *Cryptographic Hardware and Embedded Systems - CHES 2007*, pages 63–80, 2007. doi: https://doi.org/10.1007/978-3-540-74735-2_5
- [GLC+21] C. Gu, W. Liu, Y. Cui, N. Hanley, M. O’Neill, and F. Lombardi. A Flip-Flop Based Arbiter Physical Unclonable Function (APUF) Design with High Entropy and Uniqueness for FPGA Implementation. *IEEE Transactions on Emerging Topics in Computing*, 9(4):1853–1866, 2021. doi: <https://doi.org/10.1109/TETC.2019.2935465>
- [GMO14] C. Gu, J. Murphy, and M. O’Neill. A unique and robust single slice FPGA identification generator. In *2014 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1223–1226, 2014. doi: <https://doi.org/10.1109/ISCAS.2014.6865362>
- [gmp] The GNU MP Bignum Library. <https://gmplib.org/>. Accessed 6 September 2022.
- [GO15] C. Gu and M. O’Neill. Ultra-compact and robust fpga-based puf identification generator. In *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 934–937, 2015. doi: <https://doi.org/10.1109/ISCAS.2015.7168788>
- [HYKS10] 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*, pages 298–303, 2010. doi: <https://doi.org/10.1109/ReConFig.2010.24>

- [KGM⁺08] S. S. Kumar, J. Guajardo, R. Maes, G. J. Schrijen, and P. Tuyls. Extended abstract: The butterfly PUF protecting IP on every FPGA. In *2008 IEEE International Workshop on Hardware-Oriented Security and Trust*, pages 67–70, 2008. doi: <https://doi.org/10.1109/HST.2008.4559053>
- [KI15] V. P. Klybik and A. A. Ivaniuk. Use of Arbiter Physical Unclonable Function to solve identification problem of digital devices. *Automatic Control and Computer Sciences*, 49(3):139–147, May 2015. doi: <https://doi.org/10.3103/S0146411615030049>
- [KNWB11] A.R. Krishna, S. Narasimhan, X. Wang, and S. Bhunia. MECCA: A Robust Low-Overhead PUF Using Embedded Memory Array. In *Cryptographic Hardware and Embedded Systems CHES 2011*, pages 407–420, Berlin, Heidelberg, 2011. Springer. doi: https://doi.org/10.1007/978-3-642-23951-9_27
- [Lim05] D. Lim. *Extracting secret keys from integrated circuits*. M.Sc. thesis, MIT (2004).
- [LLG⁺04] J.W. Lee, D 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)*, pages 176–179, 2004. doi: <https://doi.org/10.1109/VLSIC.2004.1346548>
- [LLG⁺05] 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*, 13(10):1200–1205, 2005. doi: <https://doi.org/10.1109/TVLSI.2005.859470>
- [LP14] Y. Lao and K. K. Parhi. Statistical Analysis of MUX-Based Physical Unclonable Functions. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 33(5):649–662, 2014. doi: <https://doi.org/10.1109/TCAD.2013.2296525>

- [Mar95] G. Marsaglia. *The Marsaglia Random No. CDROM Including the Diehard Battery of Tests of Randomness; National Science Foundation: Alexandria, VA, USA, 1995.* 1995. https://en.wikipedia.org/wiki/Diehard_tests, <https://web.archive.org/web/20160125103112/http://stat.fsu.edu/pub/diehard/>
- [MCNS15] D. Mukhopadhyay, R. S. Chakraborty, P. H. Nguyen, and D. P. Sahoo. Tutorial T7: physically unclonable function: A promising security primitive for internet of things. In *28th International Conference on VLSI Design, VLSID 2015, Bangalore, India, January 3-7, 2015*, pages 14–15. IEEE Computer Society, 2015. doi: <https://doi.org/10.1109/VLSID.2015.115>
- [MJSC16] P. Méaux, A. Journault, FX. Standaert, and C. Carlet. Towards Stream Ciphers for Efficient FHE with Low-Noise Ciphertexts. In *Advances in Cryptology – EUROCRYPT 2016*, pages 311–343, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg. doi: https://doi.org/10.1007/978-3-662-49890-3_13
- [MKD10] M. Majzoobi, F. Koushanfar, and S. Devadas. FPGA PUF using programmable delay lines. In *2010 IEEE International Workshop on Information Forensics and Security*, pages 1–6, 2010. doi: <https://doi.org/10.1109/WIFS.2010.5711471>
- [MKP08] M. Majzoobi, F. Koushanfar, and M. Potkonjak. Testing Techniques for Hardware Security. pages 1 – 10, 11 2008. doi: <https://doi.org/10.1109/TEST.2008.4700636>
- [MMM⁺18] S. Maitra, B. Mandal, T. Martinsen, D. Roy, and P. Stănică. Tools in Analyzing Linear Approximation for Boolean Functions Related to FLIP. In *Progress in Cryptology – INDOCRYPT 2018*, pages 282–303. Springer International Publishing, 2018. doi: https://doi.org/10.1007/978-3-030-05378-9_16
- [MMM⁺19] S. Maitra, B. Mandal, T. Martinsen, D. Roy, and P. Stănică. Analysis on Boolean Function in a Restricted (Biased) Domain. *IEEE Transactions*

- on *Information Theory*, PP:1–1, 08 2019. doi: <https://doi.org/10.1109/TIT.2019.2932739>
- [MS02] I. Mantin and A. Shamir. A Practical Attack on Broadcast RC4. In Mitsuru Matsui, editor, *Fast Software Encryption*, pages 152–164, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg. doi: https://doi.org/10.1007/3-540-45473-X_13
- [MSE10] 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, WESS '10*, New York, NY, USA, 2010. Association for Computing Machinery. doi: <https://doi.org/10.1145/1873548.1873557>
- [MSM⁺24] M. H. Mahalat, S. Subba, A. Mondal, B. K. Sikdar, R. S. Chakraborty, and B. Sen. CAPUF: Design of a configurable circular arbiter PUF with enhanced security and hardware efficiency. *Integration*, 95:102113, 2024. doi: <https://doi.org/10.1016/j.vlsi.2023.102113>
- [MTT61] S. Muroga, I. Toda, and S. Takasu. Theory of majority decision elements. *Journal of the Franklin Institute*, 271(5):376–418, 1961. doi: [https://doi.org/10.1016/0016-0032\(61\)90702-5](https://doi.org/10.1016/0016-0032(61)90702-5)
- [MTV08] R. Maes, P. Tuyls, and I. M. Verbauwhede. Intrinsic PUFs from Flip-flops on Reconfigurable Devices. 2008. <https://api.semanticscholar.org/CorpusID:45613285>
- [Muk16] D. Mukhopadhyay. PUFs as Promising Tools for Security in Internet of Things. *IEEE Des. Test*, 33(3):103–115, 2016. doi: <https://doi.org/10.1109/MDAT.2016.2544845>
- [MYIS14] T. Machida, D. Yamamoto, M. Iwamoto, and K. Sakiyama. A new mode of operation for Arbiter PUF to improve uniqueness on FPGA. In *2014 Federated Conference on Computer Science and Information Systems*, pages 871–878, Sept 2014. doi: <https://doi.org/10.15439/2014F140>
- [MYIS15] T. Machida, D. Yamamoto, M. Iwamoto, and K. Sakiyama. Implementation of double Arbiter PUF and its performance evaluation on FPGA. In

The 20th Asia and South Pacific Design Automation Conference, pages 6–7, 2015. doi: <https://doi.org/10.1109/ASPDAC.2015.7058919>

- [MZD19] S. Mesnager, Z. Zhou, and C. Ding. On the nonlinearity of Boolean functions with restricted input. *Cryptography and Communications*, 11, 01 2019. doi: <https://doi.org/10.1007/s12095-018-0293-6>
- [PRTG02] R. Pappu, B. Recht, J. Taylor, and N. Gershenfeld. Physical One-Way Functions. *Science*, 297(5589):2026–2030, 2002. doi: <https://www.science.org/doi/abs/10.1126/science.1074376>
- [RBK10] U. Rührmair, H. Busch, and S. Katzenbeisser. *Strong PUFs: Models, Constructions, and Security Proofs*, pages 79–96. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010. doi: https://doi.org/10.1007/978-3-642-14452-3_4
- [RDK12] U. Rührmair, S. Devadas, and F. Koushanfar. Security Based on Physical Unclonability and Disorder. *Introduction to Hardware Security and Trust*, 10 2012. doi: https://doi.org/10.1007/978-1-4419-8080-9_4
- [RDM⁺23] S. Roy, D. Das, A. Mondal, M. H. Mahalat, B. Sen, and B. Sikdar. PLAKE: PUF-Based Secure Lightweight Authentication and Key Exchange Protocol for IoT. *IEEE Internet of Things Journal*, 10(10):8547–8559, 2023. doi: <https://doi.org/10.1109/JIOT.2022.3202265>
- [RRM21] A. Roy, D. Roy, and S. Maitra. How Do the Arbiter PUFs Sample the Boolean Function Class? In *Selected Areas in Cryptography - 28th International Conference, SAC 2021, Virtual Event, September 29 - October 1, 2021, Revised Selected Papers*, volume 13203 of *Lecture Notes in Computer Science*, pages 111–130. Springer, 2021. doi: https://doi.org/10.1007/978-3-030-99277-4_6
- [RSS09] U. Rührmair, J. Sölter, and F. Sehnke. On the Foundations of Physical Unclonable Functions. *IACR Cryptology ePrint Archive*, 2009:277, 01 2009. <https://api.semanticscholar.org/CorpusID:8625258>

- [RSS⁺10a] U. Rührmair, F. Sehnke, J. Sölter, G. Dror, S. Devadas, and J. Schmidhuber. Modeling Attacks on Physical Unclonable Functions. pages 237–249, 10 2010. doi: <https://doi.org/10.1145/1866307.1866335>
- [RSS⁺10b] 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, CCS '10*, page 237249, New York, NY, USA, 2010. Association for Computing Machinery. doi: <https://doi.org/10.1145/1866307.1866335>
- [RSS⁺13a] 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. *Information Forensics and Security, IEEE Transactions on*, 8:1876–1891, 11 2013. doi: <https://doi.org/10.1109/TIFS.2013.2279798>
- [RSS⁺13b] 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*, 8(11):1876–1891, 2013. doi: <https://doi.org/10.1109/TIFS.2013.2279798>
- [RUV13] B. Reichardt, F. Unger, and U. Vazirani. Classical command of quantum systems. *Nat.*, 496(7446):456–460, 2013. doi: <https://doi.org/10.1038/nature12035>
- [sag] SageMath: A free open-source mathematics software. <https://www.sagemath.org/>.
- [SBC19a] P. Santikellur, A. Bhattacharyay, and R. S. Chakraborty. Deep Learning based Model Building Attacks on Arbiter PUF Compositions. *IACR Cryptol. ePrint Arch.*, 2019:566, 2019. <https://api.semanticscholar.org/CorpusID:174783658>
- [SBC⁺19b] A. A. Siddhanti, S. Bodapati, A. Chattopadhyay, S. Maitra, D. Roy, and P. Stănică. Analysis of the Strict Avalanche Criterion in Variants of Arbiter-Based Physically Unclonable Functions. In *Progress*

in Cryptology – INDOCRYPT 2019, pages 556–577, Cham, 2019. Springer International Publishing. doi: https://doi.org/10.1007/978-3-030-35423-7_28

- [SBP⁺22] S. Singh, S. Bodapati, S. B. Patkar, R. Leupers, A. Chattopadhyay, and F. Merchant. PA-PUF: A Novel Priority Arbiter PUF. *2022 IFIP/IEEE 30th International Conference on Very Large Scale Integration (VLSI-SoC)*, pages 1–6, 2022. <https://api.semanticscholar.org/CorpusID:250921050>
- [SD07] G. E Suh and S. Devadas. Physical Unclonable Functions for Device Authentication and Secret Key Generation. In *2007 44th ACM/IEEE Design Automation Conference*, pages 9–14, 2007. doi: <https://doi.org/10.1145/1278480.1278484>
- [Sho94] P.W. Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 124–134, 1994. doi: <https://doi.org/10.1109/SFCS.1994.365700>
- [Sim94] D.R. Simon. On the power of quantum computation. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 116–123, 1994. doi: <https://doi.org/10.1109/SFCS.1994.365701>
- [SKKŠ19] M. Sýs, D. Klinec, K. Kubiček, and P. Švenda. BoolTest: The Fast Randomness Testing Strategy Based on Boolean Functions with Application to DES, 3-DES, MD5, MD6 and SHA-256. In Mohammad S. Obaidat and Enrique Cabello, editors, *E-Business and Telecommunications*, pages 123–149, Cham, 2019. Springer International Publishing. doi: https://doi.org/10.1007/978-3-030-11039-0_7
- [SKŠ17] M. Sýs, D. Klinec, and P. Švenda. The Efficient Randomness Testing using Boolean Functions. In *Proceedings of the 14th International Joint Conference on e-Business and Telecommunications - Volume 6: SECRYPT, (ICETE 2017)*, pages 92–103. INSTICC, SciTePress, 2017. doi: <https://doi.org/10.5220/0006425100920103>

- [Sor17] T. A. A. Soroceanu. *Security Analysis of Strong Physical Unclonable Functions*. PhD thesis, 2017. <https://www.mi.fu-berlin.de/inf/groups/ag-idm/theseses/2017-Soroceanu-MSc.pdf>
- [SS10] D. Suzuki and K. Shimizu. The Glitch PUF: A New Delay-PUF Architecture Exploiting Glitch Shapes. In *Cryptographic Hardware and Embedded Systems, CHES 2010*, pages 366–382. Springer Berlin Heidelberg, 2010. doi: https://doi.org/10.1007/978-3-642-15031-9_25
- [SSM⁺14] D. P. Sahoo, S. Saha, D. Mukhopadhyay, R. S. Chakraborty, and H. Kapoor. Composite PUF: A new design paradigm for Physically Unclonable Functions on FPGA. In *2014 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, pages 50–55, 2014. doi: <https://doi.org/10.1109/HST.2014.6855567>
- [SVCL18] B. Srinivasu, P. Vikramkumar, A. Chattopadhyay, and K. Y. Lam. CoLPUF : A Novel Configurable LFSR-based PUF. In *2018 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*, pages 358–361, 2018. doi: <https://doi.org/10.1109/APCCAS.2018.8605643>
- [TAB20] J. Tobisch, A. Aghaie, and G. T. Becker. Combining Optimization Objectives: New Machine-Learning Attacks on Strong PUFs. *Cryptology ePrint Archive*, Paper 2020/957, 2020. arXiv: <https://eprint.iacr.org/2020/957>.
- [Wal18] J. Walker. Pseudorandom Number Sequence Test Program. <https://www.fourmilab.ch/random/>, 2018.
- [WMP⁺20] N. Wisiol, C. Mühl, N. Pirnay, P. Nguyen, M. Margraf, J. Seifert, M. van Dijk, and U. Rhrmair. Splitting the Interpose PUF: A Novel Modeling Attack Strategy. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 97–120, 06 2020. doi: <https://doi.org/10.46586/tches.v2020.i3.97-120>
- [WTM⁺22] N. Wisiol, B. Thapaliya, K. T. Mursi, J. Seifert, and Y Zhuang. Neural Network Modeling Attacks on Arbiter-PUF-Based Designs. *IEEE Trans-*

actions on Information Forensics and Security, 17:2719–2731, 2022. doi: <https://doi.org/10.1109/TIFS.2022.3189533>

- [YI65] S. Yajima and T. Ibaraki. A Lower Bound of the Number of Threshold Functions. *IEEE Transactions on Electronic Computers*, EC-14(6):926–929, 1965. doi: <https://doi.org/10.1109/PGEC.1965.264090>
- [YSI⁺11] D. Yamamoto, K. Sakiyama, M. Iwamoto, K. Ohta, T. Ochiai, M. Takenaka, and K. Itoh. Uniqueness Enhancement of PUF Responses Based on the Locations of Random Outputting RS Latches. volume 6917, pages 390–406, 09 2011. doi: https://doi.org/10.1007/978-3-642-23951-9_26
- [ZPK⁺16] S. S. Zalivaka, A. V. Puchkov, V. P. Klybik, A. A. Ivaniuk, and C. H. Chang. Multi-valued Arbiters for quality enhancement of PUF responses on FPGA implementation. In *2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 533–538, Jan 2016. doi: <https://doi.org/10.1109/ASPDAC.2016.7428066>