Towards Security without Secrets

by

Charles Henry Herder III

Bachelor of Science in Electrical Engineering and Computer Science,
Massachusetts Institute of Technology (2009)
Bachelor of Science in Physics, Massachusetts Institute of Technology (2009)
Masters of Engineering in Electrical Engineering and Computer Science,
Massachusetts Institute of Technology (2010)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2016

Signature redacted

Author .....
Department of Electrical Engineering and Computer Science
January 29, 2016

Signature redacted

Certified by .....
Srinivas Devadas
Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Signature redacted

Accepted by .....
Leslie A. Kolodziejski
Chair, Department Committee on Graduate Students

Towards Security without Secrets

by

Charles Henry Herder III

Submitted to the Department of Electrical Engineering and Computer Science
on January 29, 2016, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Electrical Engineering and Computer Science

Abstract

Physical Unclonable Functions (PUFs) are a promising new cryptographic primitive that leverage manufacturing variation to create unclonable secrets in embedded systems. In this case, the secret is no longer stored permanently in digital form, but rather as the physical properties of the manufactured chip. Further, the recent proposal of "Public Model Physical Unclonable Functions" (PPUFs) does not contain any secrets at all. Instead, PPUFs propose to use a constant-factor computational speedup to distinguish an unclonable hardware device from a digital simulation.

This thesis presents a new computational fuzzy extractor and stateless PUF leveraging Learning Parity with Noise (LPN). This method significantly improves over the state-of-the-art in extracting stable secrets from PUFs and has a clear security reduction to a well-accepted cryptographic assumption (LPN).

In addition, this dissertation proposes for the first time a formalism describing Public Model Physical Unclonable Functions based on ordinary differential equations (ODEs), a conjecture on the form of ODE integrators, and a formal reduction of PPUF security to this conjecture. This result is extended to compare analog and digital computing more generally. Finally, this thesis provides direction for implementing a PPUF.

Thesis Supervisor: Srinivas Devadas
Title: Professor of Electrical Engineering and Computer Science

Acknowledgments

First, I would like to thank my advisor, Srini for his guidance, excitement, and steadfast commitment to scholarship over the course of this research. In addition to the material contained within this thesis, I know that I have obtained the tools required to continue to learn and study and contribute to the academic community and the field overall. I am excited to get started!

I would also like to thank Marten van Dijk for so many hours of his time discussing and solving problems in many facets of my research. I am confident that I would not have obtained many of the results contained in this thesis without our discussions, and I am thankful for the his generosity and continued support, both academically and professionally.

Next, I would like to thank Scott Aaronson for his guidance with respect to the physics of computing. His insight and experience in this field were important to establishing what computational substrates were appropriate to consider for the theoretical PPUF architecture.

I would like to thank the members of Hornet for many fun and interesting discussions both regarding my research and each of your own (and sometimes altogether unrelated to anything whatsoever). Your collective intellectual curiosity and excitement helped me sustain my own focus, especially through the many frustrating experiences that come part and parcel with any research effort.

Finally, I would like to thank my family and especially my wife Tina for her patience and support through the long hours of work. Tina, I love to share the excitement of my research and discuss all of the esoteric and minute details with you. I thank you for sharing the highs and lows of this journey with me.

Contents

List of Figures

List of Tables

1 - Introduction

Mobile and embedded devices are ubiquitous, interconnected platforms for everyday tasks. Many such tasks require the mobile device to securely authenticate and be authenticated by another party and/or securely handle private information. Indeed, smartphones have become a unified platform capable of conducting financial transactions, storing a user's secure information, acting as an authentication token for the user, and performing many other secure applications. Further, the United States has begun the deployment of smart card technologies with integrated processors and secure storage to replace magnetic stripes for authenticating a customer's credit/debit account information. The development of powerful mobile computing hardware has provided the software flexibility to enable convenient mobile data processing. However, comparable mobile hardware security has been slower to develop. Due to the inherent mobility of such devices, the threat model must include use cases where the device operates in an untrusted environment and the adversary has a degree of physical access to the system. Indeed, the adversary may in fact be the owner of the device.

The current best practice for providing such a secure memory or authentication source in such a mobile system is to place a secret key in a non-volatile EEPROM or battery-backed SRAM memory and use hardware cryptographic operations such as digital signatures or encryption. These technologies are vulnerable to invasive attack mechanisms. Protection against such attacks in current technologies requires the use of active tamper detection/prevention circuitry which has the drawback that it must be continually powered. Finally, EEPROM and other non-volatile storage technologies typically require additional fabrication steps (e.g., additional mask layers

for EEPROM).

Physical Unclonable Functions (PUFs) are a promising innovative primitive that are used for **authentication** and **secret key storage** without the requirement of secure EEPROMs and other expensive hardware described above [64, 137]. This is possible, because instead of storing secrets in digital memory, PUFs derive a secret from the physical characteristics of the integrated circuit (IC). For example, this thesis will discuss a PUF that uses the innate manufacturing variability of gate delay as a physical characteristic from which one can derive a secret. This approach is advantageous over standard secure digital storage for several reasons:

- PUF hardware uses simple digital circuits that are easy to fabricate and consume less power and area than EEPROM/RAM solutions with anti-tamper circuitry. In addition, many practical PUF applications do not require expensive cryptographic hardware such as Secure Hash Algorithm (SHA) or a public/private key encryption algorithm.

- Since the "secret" is derived from physical characteristics of the IC, the chip must be powered on for the secret to reside in digital memory. Any physical attack attempting to extract *digital* information from the chip therefore must do so while the chip is powered on.

- Invasive attacks are more difficult to execute without modifying the physical characteristics from which the secret is derived. Therefore, continually-powered active anti-tamper mechanisms are not required to secure the PUF [52].

- Non-volatile memory is more expensive to manufacture than PUF hardware. EEPROMs require additional mask layers, and battery-backed RAMs require an external always-on power source.

A PUF is based on the idea that even though the mask and manufacturing process is the same among different ICs, each IC is actually slightly different due to normal manufacturing variability. PUFs leverage this variability to derive "secret" information that is unique to the chip. In addition, due to the manufacturing variability

that defines the secret, one cannot manufacture two identical chips, even with full knowledge of the chip's design. PUF architectures exploit manufacturing variability in multiple ways. In addition to gate delay, architectures also use the power-on state of SRAM, threshold voltages, and many other physical characteristics to derive the secret.

1.1 Motivation

The two primary applications of Physical Unclonable Functions are for (1) low-cost authentication, and (2) secure key generation. These two applications have resulted from the fact that PUFs designed during the past decade have mostly fallen into two broad categories. These categories are described as "Strong PUFs" and "Physically Obfuscated Keys" (POKs).[1] Strong PUFs are typically used for authentication, while POKs are used for key storage.

Each PUF can be modeled as a black-box challenge-response system. In other words, a PUF is passed an input challenge c, and returns a response $r = f(c)$, where $f(\cdot)$ describes the input/output relations of the PUF. The black-box model is appropriate here, because the internal parameters of $f(\cdot)$ are hidden from the user since they represent the internal manufacturing variability that the PUF uses to generate a unique challenge-response set. Such parameters would include the variability of an circuit's internal gate delay as described in the introduction. PUF security relies on the difficulty of measurement or estimation of these parameters as well as the difficulty of manufacturing two chips with the same set of parameters.

The fundamental difference between POKs and strong PUFs is the domain of $f(\cdot)$, or informally, the number of unique challenges c that the PUF can process. A POK can only support a small number of challenges (in some cases only a single challenge). A strong PUF can support a large enough number of challenges such that complete determination/measurement of all challenge-response pairs within a limited timeframe is not feasible.

[1] POKs are also known as "Weak PUFs", but this thesis will refer to them as POKs.

1.1.1 POK Model

The first class of PUFs leveraging manufacturing variability are POKs. These PUFs can be thought of as PUFs that directly digitize some "fingerprint" of the circuit. This direct measurement results in a digital signature that can be used for cryptographic purposes. Because the fingerprint signature remains largely invariant, this means that the PUF can only be interrogated by one or a small number of challenges. In the above black box description, this corresponds to $f(\cdot)$ having a domain of one or only a small number of inputs. Correspondingly, $f(\cdot)$ will also have a very small range, as a given challenge should always result in the same response (ignoring noise, which is considered later). One can clearly use several instances of the above black box to support more challenge-response pairs or response bits. However, this is still considered a POK, because the number of responses is linearly related to the number of components subject to manufacturing variation. Explicitly stated, POKs have the following properties:

- Small number of challenge-response pairs (linearly related to the number of components whose behavior depends on manufacturing variation).
- Response is stable and robust to environmental conditions and multiple readings so that a challenge always yields the same response.
- Responses are unpredictable and depend strongly on the innate manufacturing variability of the device.
- It is impractical to manufacture two devices with the same physical fingerprint.

An example POK is the power-on state of an SRAM. Although an SRAM cell is symmetric, manufacturing variability will give each cell a tendency towards a logical '1' or '0' at power-on. This variability is random across the entire SRAM, giving it a unique fingerprint on power-on that can be identified. In this case, if the "response" consists of the entire SRAM state at power-on, the notion of a "challenge" is not particularly useful, as there is only one possible "challenge": powering on the SRAM. The output signature is always the same (ignoring noise). One can allow for more

output bits by increasing the size of the SRAM, but the response space is still linearly related to the number of components subject to manufacturing variation (each SRAM cell). The SRAM is an extreme example of a POK in the sense that it only has one "challenge-response pair".

Note that since POKs in general have only a small number of challenge-response pairs (CRPs), these pairs *must* be kept secret. If a POK only has one CRP, and it is revealed, then any device can emulate the PUF. For this reason, POKs are well suited for use in key derivation processes. The PUF provides the randomness and secure storage, and the secret key (derived from the PUF's response bits) is never revealed during operation.

This limitation is in tension with the fact that POKs are noisy and must be error corrected before a stable key can be produced. This is typically performed using publicly known "helper data" that is generated when the POK is provisioned. It is crucial, however, that the "helper data" used for error correction not give away too much information about the secret key.

Once the key is recovered by the PUF (this typically requires error correction), any cryptographic process may follow. For example, the POK output may be used as the key in a keyed-hash message authentication code (HMAC) challenge-response sequence. In addition, the output may be used as a secret key to encrypt/decrypt data on the device.

1.1.2 Strong PUF Model

Strong PUFs differ from POKs in that a strong PUF can support a large number of challenge-response pairs. As a result, a strong PUF can theoretically be authenticated directly without using any cryptographic hardware. The requirements for a strong PUF are:

- Large enough challenge-response space such that an adversary cannot enumerate all challenge-response pairs within a certain fixed time. (Ideally, exponential in the number of challenge bits).

- Responses stable to environment, multiple readings.
- An adversary given a polynomial-sized sample of adaptively chosen challenge-response pairs cannot predict the response to a new, randomly chosen challenge.
- Not feasible to manufacture two PUFs with the same responses.
- The readout *only* reveals the response $r = f(c)$ and no other data about the internal functionality of the PUF.

It should be noted that a POK can provide authentication capabilities if the POK is paired with cryptographic hardware supporting HMAC or similar authentication processes (note that HMAC and others support exponentially sized challenge-response spaces but their use requires 100% response stability and therefore error correction logic). It should also be noted that the security models for POKs and strong PUFs differ. The output of a POK *must* be kept private, while a strong PUF's responses do not have the same restriction.

The strong PUF has the additional requirement of readout access restriction (only $r = f(c)$ is revealed) due to this difference in security models. In addition, to prevent total enumeration of the strong PUF, one must also consider the readout time of the PUF in conjunction with the number of challenge-response pairs. A faster PUF response allows for faster enumeration of all PUF challenge-response pairs. Since a POK provides a secret key, the surrounding digital cryptographic hardware is responsible for limiting access to the POK output. However, the strong PUF does not require the use of additional cryptographic hardware to provide authentication services, and therefore must itself prevent unauthorized access into its own internal structure.

1.1.3 Limitations of POKs and Strong PUFs

In spite of the advantages that PUF systems can potentially provide to embedded systems authentication applications, there has been significant difficulty in simultaneously providing a strong security guarantee and a practical physical implementation

(cf. Section 2.4). The reason for this difficulty in each case stems from the tension between the rigid, mathematical nature of modern cryptography, and the noisy, imperfect nature of physical systems.

For POKs, this challenge manifests itself in the difficulty of reconstructing stable keys from the noisy PUF. This is referred to as Secure Sketch/Fuzzy Extraction [54]. Section 2.4 argues that in order for existing fuzzy extractors to be secure (i.e., an adversary can't compute the key even if he/she knows the helper information), they either (a) cannot correct enough errors to produce a stable key, (b) have unclear cryptographic assumptions, and/or (c) require strong, unreasonable properties in the PUF, e.g., independent, identically distributed – (i.i.d.) noise.

For strong PUFs, the physical system and the protocol are both *stateless* (i.e., store *no* data between subsequent queries and do not require non-volatile digital storage). The stateless property implies that there is no separate, irreversible "provisioning" stage: the interface exposed by the PUF is static, and any valid query can be made at any time. The simplicity and power of the above protocol motivated the construction of many candidate silicon PUFs. Unfortunately, none of the candidate constructions have a proof of computational security, and most, if not all of them have been shown to be susceptible to machine learning attacks (cf. Section 2.4). In the context of stateless PUFs, Gassend et al. [64] write: "An important direction of research is to find a circuit that is provably hard to break ...".

1.1.4 Public Model PUFs (PPUFs)

There have been investigations into next-generation PUF technology [166, 103, 135]. These works recognize several of the intrinsic limitations of PUF technology:

- For both POKs and strong PUFs, there is still a "secret" inside the PUF, although it is no longer permanently stored in digital form. For arbiter/ring oscillator PUFs, the secret is the gate delay of the inverters in the ring oscillator/arbiter chain. Therefore, it is theoretically possible that a sophisticated attacker may be able to extract these values (the gate delays) or the values

corresponding to the response of the POK to break security. Since the PUF's secret cannot be published without compromising security, the PUF protocols are limited to "symmetric key" applications.

- A strong PUF implementation requires the verifier to memorize a list of challenge/response pairs that can only be used once. This is more inefficient and cumbersome than a protocol where the verifier could generate his/her own challenges for the PUF.

As a result, the "Public Model Physical Unclonable Function" has been proposed (cf. Chapter 5 for more details). Loosely speaking, this architecture could enable a "public-key" variant of a physical unclonable function that also allows a verifier to compute his/her own challenges. This is in direct tension with the requirement for standard PUFs that the manufacturing variation of the PUF remain secret, so that the behavior of the PUF cannot be duplicated. Therefore, there is more to the PPUF protocol, discussed below.

In the PPUF construction, there are *no secrets* anywhere in the device. Authentication is based on (1) the unclonability of the PPUF hardware device, and (2) a computational speedup of the PPUF hardware device over any digital model.

The digital model associated with the PPUF hardware is publicly known. However, the timing difference between the model and the PPUF hardware is used as a source of secure authentication. The protocol is shown in Figure 1-1 [107, 130].

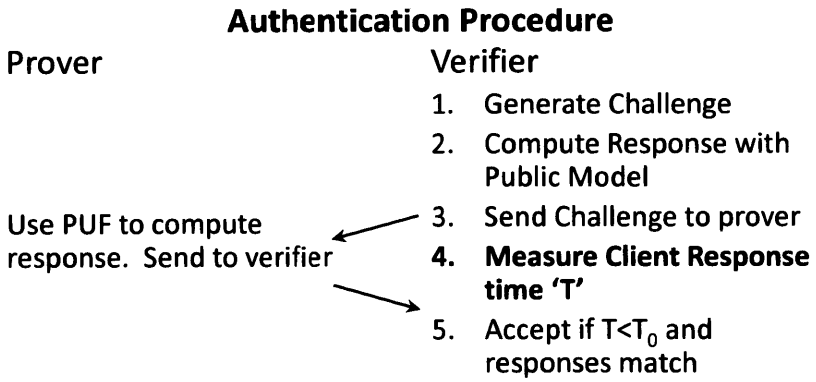


Figure 1-1: Authenticating a Public Model PUF system.

As a result, only the person in possession of the device is able to authenticate. Since there are no secrets to steal, the security guarantee can be much higher than modern secure hardware. This "security without secrets" has application to numerous industries. For example, using a PPUF system on a credit card would bind unequivocally account access to the possession of the card.

Today, credit card information theft is a major problem, with millions of identities stolen per year. Billions of dollars are spent each year on the prevention of breaches and the aftermath of breaches where credit card information is stolen. The use of PPUF technology would eliminate the threat of loss of credit card information, as only theft of the card itself would allow someone to access an account. Furthermore, if such a card were ever recovered after being lost, the security would remain intact for future account access, as an adversary would not have been able to duplicate the card. Only the person in possession of the card would be able to authenticate an online purchase, for example.

1.1.5 Limitations of Public Model Physical Unclonable Functions

The concept of "security without secrets" has existed for some time, with many PPUF architectures that have been proposed [108, 104, 16, 130]. However, to date, proposed PPUF architectures do not have security guarantees that can be strongly argued, much less a formal security guarantee and reduction to a reasonable mathematical conjecture. As a result of this lack of formalism, although "security without secrets" is a groundbreaking concept, its potential has not been explored by the cryptographic community.

For example, Majzoobi, Nably, and Koushanfar presented a time-bounded authentication in 2010 that uses FPGA cells as the delay elements. The security argument is simply that it takes significantly more time to simulate the FPGA in software than it does to actually run the FPGA [103]. This security trivially breaks if the adversary has an FPGA.

Another example, Cellular Non-linear Networks (CNNs) used as a part of the proposed SIMPL systems [130] consist of an array of analog cells, that each behave according to some linear ODE that is connected to its neighboring cells. Once again, the speedups are only argued for software adversaries. The CNN can be digitally simulated very fast in parallel on an FPGA by leveraging the intrinsic parallelism of the problem. No argument is given for speedup over an adversary with sufficient FPGA resources, much less a custom mixed-signal ASIC. This is not consistent with the fact that the CNN proposal requires custom mixed signal hardware.

Therefore, in each case there is a assumption that the honest party is more powerful than the adversary – in both cases, the adversary is software only. This is in general not a useful threat model.

1.2 Contributions

1.2.1 Physically Obfuscated Keys

In this work, Learning Parity with Noise (LPN) (cf. Section 3.1) is used to provide a computationally secure, practical fuzzy extractor construction that improves substantially over the state of the art. The construction is integrated into existing PUF formalism [10, 58], and provides a security reduction from the definition of a fuzzy extractor to the hardness of LPN. Further, the theory of LPN is extended to understand what types of PUF distributions are allowed. The requirements on the PUF bit distribution for the construction is stronger than a min-entropy requirement, but significantly weaker than an i.i.d. noise requirement.

Further, the construction uses LPN in a unique, novel way. I explore the intuition for why LPN provides the correct mathematical structure for efficient fuzzy extraction. Further, the parameter choices for an LPN fuzzy extractor significantly increases the difficulty of the LPN problem compared with more traditional LPN constructions. I use the best known attacks to construct a security parameter for choices of LPN fuzzy extractor parameters.

Using the parameters for a security parameter of 128, I use experimental data collected from ring-oscillator PUFs implemented on a Xilinx FPGA to verify the efficiency of the construction. A substantial improvement is demonstrated for the LPN construction in several figures of merit over the state-of-the-art.

1.2.2 Strong PUFs

Next, using the LPN fuzzy extractor, I provide a construction for a computationally secure stateless, strong PUF. The construction is provided in the context of existing PUF formalism [10]. It is proven that the construction is strongly unpredictable, and thus not vulnerable to any machine learning attacks (as most, if not all previous stateless PUF constructions are). I show a security reduction to standard LPN with an i.i.d. noise assumption on the PUF. I also show a security reduction to an LPN-variant with a weaker noise assumption on the PUF.

The computationally secure construction of a stateless Physical Unclonable Function in this thesis is based on precise hardness assumptions. This has been an open problem for over thirteen years since silicon PUFs were introduced in 2002 [64].

1.2.3 Theory for Public Model Physical Unclonable Functions

This work first identified that a formal approach to security was needed to enable future study of PPUFs. This work proposes for the first time a formalism describing Public Model Physical Unclonable Functions in terms of the relative speedup of one computational modality over another. Using this formalism, this work provides a definition of security and a formal PPUF construction (cf. Section 8.5).

These definitions represent a theoretical foundation on which the emerging technology of Public Model Physical Unclonable Functions may be studied. This foundation can be viewed as an extension of existing PUF formalism [10] (cf. Section 8.1).

Next, I present a mathematical conjecture (cf. Section 8.2) regarding the form of

numerical integration algorithms alongside an empirical justification of this conjecture in the context of previous work in the field of numerical integration. I present a reduction from the security guarantee for the PPUF construction to the conjecture (cf. Section 8.5). Finally, I provide recommendations for the physical implementation of a PPUF system.

The development of the theory and conjecture required the study of multiple fields spanning classical mechanics, complex analysis, differential geometry, symbolic computation, and complexity theory. The final theory and conjecture presented in this thesis draw from Information Theory, Differential Galois Theory, Padé Approximation, and more general asymptotic approximation theory.

The theoretical foundation presented in this thesis is necessary to enable further development in the field of Public Model Physical Unclonable Functions, and this work establishes that potential exists in the field of PPUF systems to enable "security without secrets."

1.2.4 Theory for Complexity of Analog Computing vs. Digital Computing

In Section 7, I recognize that the key challenge for PPUF implementation is an instance of a broader complexity theoretic question regarding the relationship between analog and digital computing. Over the course of providing the security reduction for the PPUF construction, I also recognize several implications for the more general field of analog computing.

In particular, there has been continuing interest in finding applications where analog computing has a speed advantage over digital computing [111, 24, 152, 122, 45]. In this work, I show a particular mathematical problem wherein analog computing does have a circuit complexity advantage over any digital simulator (assuming the conjecture mentioned above). This result provides the first constant-factor separation between analog and digital computing modalities under a precise and plausible conjecture.

2 - Physical Unclonable Functions Background and Related Work

Although many of the architectures that integrate PUFs into existing IC technology are recent, it should be noted that the concepts of unclonability and uniqueness have been used extensively in the past for other applications [86]. For example, "Unique Objects" are well defined as objects with a unique set of properties (a 'fingerprint') based on the unique disorder of the object [137]. One example of early usage of unique objects for security was proposed for the identification of nuclear weapons during the cold war [67]. One would spray a thin coating of randomly distributed light-reflecting particles onto the surface of the nuclear weapon. Since these particles are randomly distributed, the resulting interference pattern after being illuminated from various angles is unique and difficult to reproduce. Unique objects were termed Physical One-Way Functions and popularized in 2001 [121]. However, to the author's knowledge, none of these proposals has an associated computational security argument that shows hardness of model-building or machine learning attacks.

There are two categories of Physical Unclonable Functions (PUFs) that are used in modern embedded cryptosystems: POKs and strong PUFs as described in Section 1.1. POKs have one or a small number of challenge/response pairs, while strong PUFs have a large (generally exponential in the size of the system) challenge response space.

This chapter discusses how both POKs and strong PUFs are implemented, and where the current implementation and theoretical challenges lie.

2.1 Example Strong PUF Architectures

2.1.1 Optical PUF

One of the first implementations of a strong PUF was constructed by Pappu et al. in 2001 [120]. The paper terms the device a "Physical One-way Function," but the functionality is identical to that of a strong PUF. Pappu describes a device with three primary components: (1) A laser directed along the Z axis that can be moved in the XY plane and whose polarization can be modified, (2) A stationary scattering medium that sits along the path of the laser beam, and (3) An imaging device that records the output 'speckle' pattern of laser light exiting the scattering medium.

In this device, the input challenge is a laser XY location and polarization, and the response is the associated speckle pattern. The speckle pattern is strongly dependent on the input location/polarization because multiple scattering events occur inside of the scattering medium. In the implementation by Pappu et al., the scattering medium consisted of a large number of randomly positioned 100 µm silica spheres suspended in a hardened epoxy. Each sphere acts as a small lens, refracting individual rays of light as they move through the scattering block. The overall size of the scattering block was on the order of 1mm thick. Therefore, even a relatively simple optical path must encounter ~10 spheres as it travels through the scattering block.

All of these paths then are focused into an image on the detector. It is intuitively true that each of these paths will be very sensitive to input coordinates. Studies on speckle patterns produced by reflection/transmission by rough surfaces have found this to be true both experimentally and mathematically [43]. In addition, the speckle pattern is also sensitive to the internal structure of the scattering block. Therefore, it is difficult to fabricate two blocks with identical speckle patterns. Finally, due to the complex nature of the physical interactions, it is difficult to model the internal dynamics of the scattering medium. It is also difficult to use the output speckle to determine properties of the scattering block (such as the locations of the silica spheres).

These assumptions, while not strictly based on known computationally difficult

problems, are thought to be difficult due to the fact that ray-tracing electromagnetic simulation is a well-studied field with established theoretical models and best practices. One can make the statement that if an adversary were able to break the above optical PUF by efficiently reproducing the physical device, modelling the entire scattering block, or discovering the sphere locations via observation of the speckle pattern, this would probably represent a major advancement in the field of ray-based models of electromagnetic simulation. It is for this reason that Pappu et al. described this Optical PUF as a "Physical One-way Function". However, there is no formal hardness assumption or reduction.

2.1.2 Arbiter PUF

Although the capabilities of the above optical PUF are significant, and they represented a significant step forward in the understanding and construction of PUFs, the practical applications are limited due to the macroscopic optical nature. This limitation stemmed from two properties.

First, the actual unclonable object (the scattering block) was *separate* from the measurement apparatus (the imaging device). As a result, the trust gained from authenticating an optical PUF is more limited. In a practical use case, the objective of authenticating the PUF is typically to authenticate the associated processor to which it is connected. However, since the optical PUF is separated from the digital measurement circuitry, an optical PUF as described by Pappu et al. designed to authenticate Processor A can easily be detached from Processor A and connected to Processor B. Processor B could then authenticate itself as Processor A. It is more desirable for the digital measurement apparatus to be integrated in with the PUF such that the PUF is not separable from the device it is used to authenticate.

Second, since both key generation and authentication applications use integrated electronics, a more practical PUF would have the same properties as the optical PUF and simultaneously be integrated directly with a conventional CMOS process. This integration would be such that the integrated circuit could not be separated from the PUF.

Silicon implementations of candidate strong PUFs were described in papers by by Gassend et al. beginning in 2002 using manufacturing variability in gate delay as the source of unclonable randomness [64]. In one implementation, a race condition is established in a symmetric circuit. This is shown in Figure 2-1. An input edge is split to two multiplexors (muxes). Depending on the input challenge bits ($X[0] - X[127]$), this path will vary. Although the layout is identical (propagation time should be the same for each edge no matter what challenge bits are chosen), manufacturing variability in the gate delay of each mux will result in one edge arriving at the latch first, and the latch acts as the 'arbiter'. The output will therefore depend on the challenge bits.

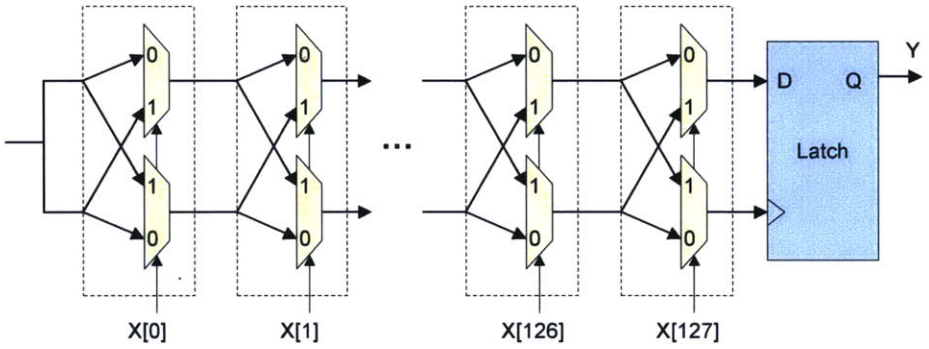


Figure 2-1: An arbiter PUF circuit. The circuit creates two delay paths with the same layout length for each input X, and produces an output Y based on which path is faster.

In Figure 2-1, there are 128 challenge bits and 1 response bit. Of course, one can operate multiple identical circuits in parallel to achieve 128 response bits. In this way, the arbiter PUF can be scaled to an almost arbitrary number of challenge-response pairs.

The security of the arbiter PUF, like the optical PUF before it, is based on assumptions regarding manufacturing capabilities and ultimately metrology of the individual gate delays. Because the design is symmetric, the design does not contain any 'secret' information. An adversarial manufacturer that has the PUF design cannot manufacture a duplicate PUF, because the behavior of the PUF is defined by the inherent variability in the manufacturing process. Even the original manufacturer of

the PUF could not produce two identical PUFs, since this would require a significant improvement in manufacturing control.

The second security assumption is that the individual gate delays are difficult to measure directly. It assumes that an invasive attacker would have difficulty in extracting the individual delays even with physical access. This assumption is based on the hypothesis that an invasive attacker would destroy the gate delay properties using his/her measurement techniques.

The last security assumption is that given a set of challenge-response pairs from an arbiter PUF, an adversary could not calculate the internal delays of the gates. For the architecture described above, this is actually not the case. Each delay is independent from all other delays, and the delays add linearly. As a result, one can use standard linear systems analysis to intelligently gather data about the gate delays from the response bits. In fact, it can be shown that this system breaks after only a small number of challenges [96]. This problem can be partially mitigated by several approaches proposed by Gassend et al. and described in Section 2.2.4.

Finally, environmental factors play a significant role in both optical and arbiter PUF architectures. For the optical PUF, calibration of the input location is a concern. In the case of the arbiter PUF, environmental variations such as temperature, supply voltage, aging, and even random noise will affect the delay of each edge through the arbiter PUF. In addition, if the delays are close enough, the latch's setup time will be violated, potentially resulting in an unpredictable output. As a result, the response bits may not be stable. The impact of noise is is mitigated in the case of the arbiter PUF by the differential nature of the construction (cf. Figure 2-1), though not completely eliminated. In this case, error correcting techniques are used to increase the stability of the PUF while maintaining its security. Techniques for accomplishing this will be covered in Section 2.2.5. Although key generation has zero error tolerance, PUF authentication usually incorporates an allowable error threshold, thereby decreasing the stability requirement, and often obviating the need for error correction.

2.2 Low-Cost Cryptographic Authentication: Strong PUFs

The strong PUF architectures described above are typically associated with the application of low-cost authentication. In this case, a strong PUF will replace the secure memory and cryptographic hardware on an embedded device and is used to securely identify the device to a server. Because the PUF does not require secure non-volatile memory, anti-tamper circuitry, or additional supporting cryptographic acceleration hardware, a PUF-based solution requires less area, power, and mask layers than a traditional approach to secure authentication.

2.2.1 Authentication Protocol

As described previously, the strong PUF receives a challenge and generates a response. However, the requirements of a strong PUF state that an adversary provided with polynomial challenge-response pairs should not be able to predict the response to a new challenge.

Although this is a desirable property, it also presents a usage problem. Since the PUF acts as a "black box," even the authentication server only has access to previously observed challenge-response pairs and therefore also cannot predict the response to a new challenge.

Therefore, the protocol for using PUFs is significantly different than most public/private key cryptographic systems. Consider a **server** authenticating a **client** in Algorithm 1.

Because the server cannot predict the PUF behavior, it must internally store challenge-response pairs to be used later. Each challenge-response pair must be used *only once*. Therefore, the server must either store enough challenge-response pairs so that it will not run out, or it must periodically "recharge" the table by establishing secure communication with an authenticated client and requesting responses to new challenges. To address the CRP table scalability problem, newer protocols based on

Algorithm 1 Protocol for authenticating a Public Model PUF.

1: **PUF** is manufactured.
2: **Server** obtains access to **PUF** and generates a table of challenge-response pairs. These pairs are stored in internal secret storage.
3: **PUF** is given to **client**.
4: **Client** submits a request to the **server** to authenticate.
5: **Server** picks a known challenge-response pair and submits the challenge to the **client**.
6: **Client** runs the challenge on the PUF, returns the response to the **server**.
7: **Server** checks to see that the response is correct and marks the challenge-response pair as *used*.

storage of a compact model for PUF have emerged [107, 51, 104, 89].

Note that each client PUF will have unique challenge-response pairs, and therefore can be individually authenticated. In addition, the server must store tables of challenge-response pairs for *each* of the clients to be authenticated.

2.2.2 Arbiter PUF Topologies

The initial implementation of silicon PUFs had known security issues due to the fact that the delays were linearly added to produce the resultant response bit [61]. As a result, they could be learned with relative ease. This issue naturally led to the introduction of other "nonlinear" effects to make such modelling attacks more difficult. These efforts included XOR arbiter PUFs, Lightweight Secure PUFs, and Feedforward arbiter PUFs [158, 93, 96, 61, 105].

In an XOR Arbiter, multiple arbiter PUF outputs are XOR'ed to form a single response bit. This is shown in Figure 2-2. These structures have shown greater resilience against machine learning attacks [131, 107]. However, recent studies have demonstrated the vulnerability of the XOR Arbiters to a combination of machine learning and side-channel attacks [138, 102, 139]. Developing methods to suppress the side-channels could help in alleviating this vulnerability.

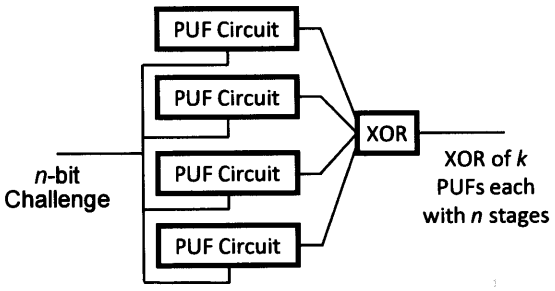


Figure 2-2: Four individual arbiter PUF circuits with nonlinearities introduced via XOR'ing their outputs.

2.2.3 Arbiter PUF Implementation

The arbiter PUF was first implemented and studied by Lim et al. [96]. Devadas et al. also implemented arbiter PUFs as a part of a RFID IC fabricated in 0.18 µm technology [52]. In this implementation, a single arbiter PUF is implemented on-chip. This primitive has an input challenge of 64 bits and a single output bit. To construct a k-bit response, a linear feedback shift register (LFSR) is used to generate a pseudorandom sequence based on the input challenge. The PUF is then evaluated k times using k different bit vectors from this larger pseudorandom sequence. Finally, to prevent learning attacks on the PUF output bits, an additional scrambling routine is performed.

In this implementation, area and power consumption represented a major design constraint. Therefore, the above PUF implementation with only a single arbiter is used. As a result, the majority of the silicon area is consumed by standard RFID components (RFID front-end, one-time programmable memory, digital logic). The PUF and associated LFSR have been implemented in less than 0.02mm^2 using 0.18 µm fabrication technology. In addition, the PUF only consumes dynamic power during evaluation, and the power consumption was shown to be small with respect to the power stored on the RFID chip.

In order to understand the PUF's utility as an identification and authentication source, **Intra-PUF** and **Inter-PUF** variation are defined as follows [64]:

- **Intra-PUF Variation**: Defined as the number of bits in a PUF response that

vary when an identical challenge is repeatedly queried on a given PUF device in a changing environment. This variation is due to this environmental change as well as statistical noise. As a result, it is commonly represented in the form of a statistical distribution. Intra-PUF Variation is a measure of the reproducibility of responses from an individual PUF circuit.

- **Inter-PUF Variation**: Defined as the number of bits in a PUF response that vary between different devices for a set of shared challenges. This is due to differences between the physical ICs and is also commonly represented in the form of a statistical distribution. The Inter-PUF Variation is a measure of the uniqueness of an individual PUF circuit.

For the application of secure authentication, Intra-PUF variation should be low (ideally 0%) so that the PUF can be verified. On the other hand, Inter-PUF variation should be high (ideally 50% on average) so that two separate PUFs have a maximally decorrelated responses. This behavior has been observed for arbiter PUFs as shown in Figure 2-3. Note that in this figure, the two Inter-chip variations are roughly 50% (128 bits out of 256), and the Intra-chip variations are much smaller ($\sim$10%). Clearly, the implemented PUF has the desired properties.

For key generation applications, error correction must be used to compensate for the $\sim$10% Intra-chip variation. However, for authentication applications, this error may be forgiven by observing whether the Hamming distance between the recorded PUF response and the new PUF response (cf. Algorithm 1) is closer to 50% or 10%.

2.2.4 Attacks on Arbiter PUFs

The security of a strong PUF depends on several factors. If any one of these factors is compromised, the security of the PUF itself is also compromised.

- Difficulty of measurement of PUF internal parameters (only challenge-response pairs can be measured).
- Difficulty of manufacturing "clones".

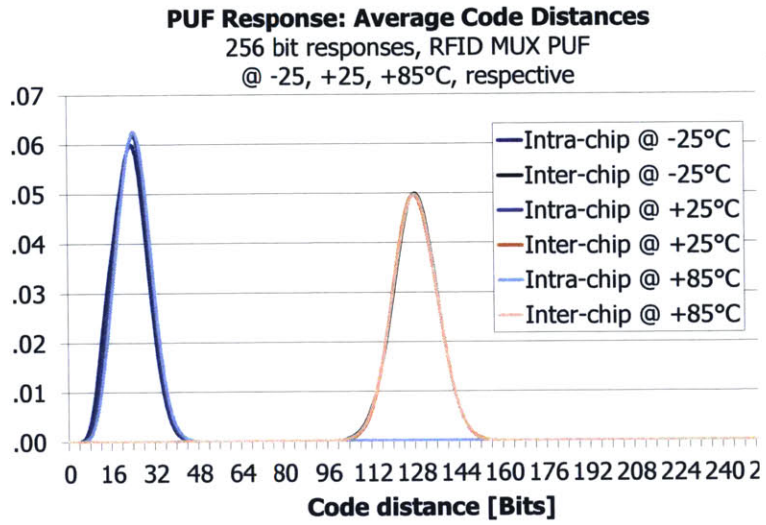


Figure 2-3: Code distance distribution for 256-bit PUF responses.

- Difficulty of predicting PUF behavior based on past challenge-response pairs.

For arbiter PUFs, an area of continued active research is in ensuring the difficulty of predicting PUF behavior based on past challenge-response pairs. In the RFID IC example, this issue is addressed by "scrambling" the output bits of the PUF. In other words, the output bits of the PUF pass through some digital circuit that obfuscates the linear behavior of the PUF before being returned as a response. The simple arbiter PUF implementation without output post-processing is linear and therefore significantly easier to predict. (Note that this scrambling will increase the noise in the output bits and therefore has to be done carefully.)

Studies have been performed by Rührmair et al. [131, 139] and Majzoobi et al. [106] using machine learning to predict the behavior of PUFs after a certain number of challenge-response pairs have been observed. In these studies, learning attacks were perpetrated on simple arbiter PUFs, feedforward arbiter PUFs, arbiter PUFs with output XOR'ing, "lightweight secure" arbiter PUFs (these use a more complicated output postprocessing circuit, but are based on principles similar to output XOR'ing).

The linear behavior of simple arbiter PUFs was clearly demonstrated, as a learning algorithm predicted the behavior of a 64-bit arbiter PUF with 95% accuracy after observing 640 challenge-response pairs (the model training time on a standard PC

was 0.01 sec). To predict with 99.9% accuracy, 18,050 challenge-response pairs needed to be observed (model training time of 0.6 sec). This demonstrates that the behavior of a simple arbiter PUF can be learned efficiently [131].

In addition, PUFs with 64-bit challenges and 128-bit challenges were tested. It was found that the number of challenge-response pairs and model learning complexity scaled as expected with the input challenge size. This proved to be true not just for simple arbiter PUFs, but also for the nonlinear implementations discussed below.

There are multiple proposals to bolster the arbiter PUF [65] architecture to prevent modeling, such as feedforward Arbiter [95] and XOR arbiter PUF [158]. Machine learning attacks such as those of [131, 139] and [14] have successfully attacked these constructions to create software clones.

A newer set of attacks leveraging both machine learning and side-channel information has recently emerged [138, 102, 139, 59]. It has been shown that by coordinated application of timing or power side-channel analysis and adapted machine learning techniques, very efficient attacks can be performed, i.e., attacks that use linearly many CRPs and low degree polynomial computation times. The practical viability of the combined attacks has been demonstrated by machine learning experiments on numerically simulated CRPs. This work has shown that XOR arbiter PUF and Lightweight PUFs have to be implemented in such a manner that power side channels are protected, else PUFs can be easily cloned.

Finally, the work by [14] shows that XOR PUFs can be broken regardless of the XOR fan-in (the value k in Figure 2-2). The key insight was their use of *reliability* instead of *PUF output* as the parameter for the machine learning fitness function. The XOR gates add non-linearity to the PUF output bits, which increases difficulty for machine learning algorithms. However, consider that each PUF Circuit in Figure 2-2 has a "reliability" for a given challenge – a probability of error. This probability of error *adds* to the probability of error for the other PUF Circuits to produce the overall probability of error. This is a *linear* relationship, so one can solve for the reliability of PUF Circuit 1 for a given challenge by repeatedly providing the same challenge for PUF Circuit 1, but varying the challenge for PUF Circuits $2 \ldots k$. In this

way, [14] shows that one can learn the parameters of each PUF circuit independently, and increasing k does not increase the security of the PUF.

The attack in [14] does require the PUF to process multiple identical challenges in order to measure reliability. Therefore, this attack may be thwarted if the PUF records received challenges and only responds to a challenge *once*. However, such a secure, non-volatile storage is by definition incompatible to the stateless nature of the definition of a PUF.

In addition to the feedforward and XOR arbiter PUF constructions, there are other constructions using nonlinear circuit elements (e.g., [96], [92], [116]) that have not yet been broken to the author's knowledge. However, these constructions do not as yet have clear computational security reductions.

One key consideration in studying the complexity of PUFs is stability (described as "Intra-PUF variation" in the RFID IC example). This will be discussed further in Section 2.2.5. However, it should be noted that although output XOR'ing has an exponential effect on modelling complexity, it also has an exponential effect on decreasing stability. In doing so, it decreases the effectiveness of a PUF in an actual authentication environment and simultaneously decreases the accuracy requirement of an attack model, as the greater intrinsic PUF error must be tolerated by the authentication protocol.

The task of identifying an approach to exponentially increase model complexity while only having a polynomial effect decreasing PUF stability is an area of active research.

2.2.5 Error Correction versus Tolerance

In the RFID IC example, random noise contributes to the PUF stability being roughly 90%, i.e., the Intra-PUF variation is ~10% . In addition, this stability worsens when the temperature changes.

In perhaps the earliest reference to error correction in silicon PUFs, Gassend mentioned the use of 2D Hamming codes [62]. Suh et al. suggested the use of Bose-Chaudhury-Hochquenghen (BCH) codes – more specifically the BCH (255,63,t = 30)

code [157]. In this case, the PUF generates 255 bits, but the code exposes 192 syndrome bits publicly, so the actual security of the system is at most 63 bits. This code corrects at most 30 errors out of 255 bits. This corresponds to a PUF with ~88% stability.

In low-cost authentication applications, the host instead gives a certain error tolerance or multiple authentication opportunities to a PUF before rejecting the PUF as invalid. Error tolerance is typically the preferred methodology. Using the arbiter PUF described with code distances shown in Figure 2-3, false positive/negative identification probabilities were measured for specific allowed error tolerances. These data are summarized in Figure 2-4.

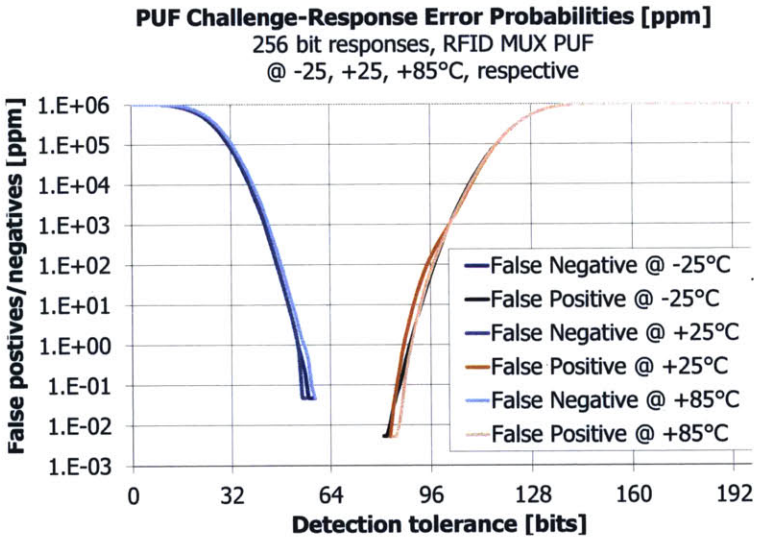


Figure 2-4: False positives and negatives for strong PUF operation with a given error tolerance.

2.3 Cryptographic Key Generation: Physically Obfuscated Keys (POKs)

Due to their limited challenge-response space, POKs are typically used for cryptographic key generation. In this case, a POK will replace a secure non-volatile memory that would have stored the cryptographic key. Once the key is derived from the POK,

it is stored in secure volatile memory during the device's operation. This key can then be used for authentication, encryption, and other cryptographic protocols. Due to the fact that one or very few keys can be generated by the PUF, the security of this key during operation is of paramount importance. If the secure key is revealed, any device can emulate the POK.

2.3.1 Key Generation Protocol

Because POKs like the ones discussed above have effectively fixed "challenge bits", the key generation protocol is fairly simple. In the case of the SRAM PUF, one simply powers on the SRAM and observes the memory state. Similarly for the ring oscillator PUF, one simply pair-wise compares each of the oscillators in order to measure the correct ordering of oscillation frequency.

In both of these cases, the complexity lies in the limitations of physical implementations that result in both statistical and systematic noise that must be corrected/mitigated. The actual approach used to address these issues differs for SRAM and ring oscillator PUFs because the underlying physical implementation is different.

Ultimately, a stable set of unique bits is extracted from the POK. These bits can then be used in any of a number of cryptographic protocols. Note that POKs can be used for authentication (similar to strong PUFs) even though they do not have a large number of challenge-response pairs. By supplementing the POK with a hardware HMAC/AES implementation, one can achieve authentication capability at the cost of the additional power and area required by the cryptographic hardware primitives that embody the HMAC/AES protocol.

2.3.2 SRAM PUF Implementation

As previously mentioned, the SRAM PUF leverages the threshold voltage mismatch of transistors in a SRAM cell due to manufacturing mismatch. This mismatch results in a repeatable tendency to settle into a '1' or '0' state when the SRAM cell is powered on with no writes occurring. Several studies have constructed SRAM PUFs

and analyzed their properties.

One of the first implementations of such a chip identification system was tested by Su et al. with RFID applications [156]. In this study, a custom SRAM cell was constructed to minimize potential systematic mismatch between the two transistors. Such a skew would result in a given SRAM cell being more likely to favor a '1' than a '0' or vice versa, even with random process variation. To prevent such systematic skew, they used analog layout techniques to construct a 'symmetric' and 'common centroid' layout of the SRAM cell.

The study demonstrated that the SRAM PUF behaved as desired. After fabrication, an equal number of SRAM cells tended towards '1' and '0' to within experimental error for both layouts. The study identified that cell positioning within the SRAM, SRAM positioning on the wafer, and subsequent wafers were all decorrelated with the SRAM cell's tendency towards '1' or '0'.

A challenge arose with the recognition that roughly 4% of the SRAM cells did not have enough mismatch to strongly favor '1' or '0'. These cells probabilistically settled into '1' or '0' at random due to the contributions of thermal and shot noise. The number of these unstable bits increased at temperature/voltage corners and as the chip aged.

The study by Holcomb et al. tested the functionality of SRAM PUFs on off-the-shelf RAM and processor products such as the MSP430 and Intel's WISP RFID device [74]. In this application, the SRAM cell was a part of another on-chip SRAM that was actively used for program/data and *not* custom fabricated in any way to enhance stability or skew performance.

In this way, an end user can use an existing off-the-shelf component with no silicon modification and, using software alone, implement a POK for cryptographic or identification purposes.

Although off-the-shelf SRAM cells are not optimized for usage as a PUF, Holcomb et al. did observe a bit stability of 5% across temperatures from 0°C-50°C. This stability is roughly the same as the stability measured by Su et al., indicating that the custom fabrication did not help significantly in this regard.

However, the off-the-shelf SRAM cells were observed to have a significant bias towards the '1' state. This changes the entropy and unique identification analysis. In this study, 512 bytes of SRAM were used as a fingerprint. Due to the systematic skew of the SRAM cells, the min-entropy of this block was roughly 200 bits (plus/minus 10 bits depending on temperature). This 512-byte block was then passed through a universal hash to extract 128 bits of output data.

Finally, because the SRAM is being used as a memory element for the processor, it is always powered, even if the PUF section of the SRAM is never written during normal operation. This continual powering of the SRAM in a '1' or '0' state results in Negative Bias Temperature Instability (NBTI). This is a type of 'burn-in' for deep submicron MOSFET technology, where the threshold voltage of a transistor increases over time due to the applied stress conditions of high temperature and a constant vertical electric field across the gate terminal while the transistor is 'on'.

Therefore, if an SRAM cell is powered on and set to the '0' state for a long time ($\sim$ 10 days), then on subsequent power-on sequences, the cell is more likely to skew towards the '1' state. This predictable behavior stands in contrast to the behavior of temperature variations, which can either skew the cell towards '0' or '1' as the temperature fluctuates.

Attacks on SRAM PUFs

Because the SRAM PUF provides a secure key (as opposed to providing challenge-response functionality like the strong PUF), it relies on other conventional security primitives to keep that key protected while the chip is powered. As a result, any side-channel or other vulnerabilities associated with the cryptographic hardware pose a threat to the secret key outputted by the SRAM PUF. In addition, since this key is kept secret, the modelling attacks used against strong PUFs cannot be used, since no input-output relations of the PUF should ever be revealed.

However, there are other ways identified in the literature to attack a SRAM PUF more directly. Many of these depend on the level of access that one has to the SRAM. If one can insert a 'write' command, then one could leverage the NBTI to

deliberately force individual bits towards '1'. If one could modify the temperature, one could potentially cause the PUF to fail by running the PUF outside of its design area. Finally, the ability for a SRAM cell to maintain its state depends on the supply voltage. If, during the turn-on process, the supply voltage is held for some time at a low ($\sim$ 100mV) voltage, the thermal noise will induce a transition into the cell's favored state, resulting in higher stability. However, if the voltage turn-on is fast, then cells become less stable. An attacker with access to the power channel could potentially control the stability of some of the SRAM PUF bits through this mechanism [75].

Recently, it has also been identified by Helfmeier et al. that the SRAM power-on state can be observed via near-infrared imaging of the SRAM during the turn-on transient. Once the SRAM "fingerprint" has been measured (the PUF response bits have been stolen), one can use focused ion beam (FIB) techniques to modify a second IC to have a matching fingerprint as the first by cutting traces and/or demolishing transistors in the SRAM cell [69].

Finally, one notes that SRAM data is not erased immediately on power down. The data remains 'stored' in the SRAM cell for a certain short time after the cell is powered down due to an effect called 'data remanence'. Oren et al. have demonstrated that this effect can be used to inject faults into the SRAM PUF. In doing so, one may be able to non-invasively learn the SRAM PUF output bits indirectly [115].

2.3.3 Ring Oscillator PUF Implementation

Yu and Devadas designed a delay-based POK based on the ring oscillator architecture, and proposed the first PUF key generation architecture that does not require traditional error correction [174, 173, 176]. The proposed Index-Based Syndrome Coding method is a departure from prior error correction schemes based on Code-offset Syndrome [54], where the syndrome format enables soft decision functionality without the complexities associated with an explicit traditional soft decision error correction decoder, which in general has a higher complexity than an equivalent hard decision error correction decoder.

In this architecture, several oscillator PUF banks are instantiated, with each oscillator bank comprising of $2k$ ring oscillators. A k-bit challenge is applied to each bank, to determine which oscillators correspond to the top delays, and which oscillators correspond to the bottom delays. The top and bottom rows are summed to produce x and y respectively. These values are used to produce a single bit PUF output and associated "soft-decision" information corresponding to a PUF challenge. Specifically, the output bit is the sign of $x - y$. The 'confidence' (discussed more in Chapters 3 and 4) is derived from the magnitude of $x - y$.

Figure 3-1 shows a simplified diagram for illustrative purposes. More complex "recombination" functions using XORs or amplitude modulation based on additional challenge bits were used in actual implementation.

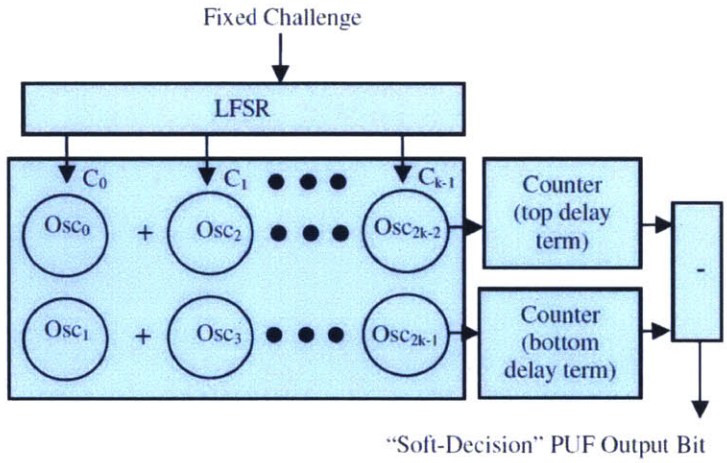


Figure 2-5: A k-sum ring-oscillator PUF. Ring oscillators that are closer in frequency do not affect the output bit due to the summation process.

Each of the oscillators is configured with 'challenge bits'. For the purpose of cryptographic key generation, these bits are fixed (see the 'Fixed Challenge' in Figure 3-1) in order to reproduce the same key each time.

Yu et al. designed multiple circuit topologies to implement this technique using a 0.13 µm CMOS process [173]. They found that, as expected, these devices produced output bits that passed all NIST standard randomness tests, demonstrated close to ideal decorrelation between different PUF devices, had a worst-case bit bias less than 0.5%, and a raw intra-device variation of ~10%.

Attacks on Ring Oscillator PUFs

Like the SRAM PUF, the ring oscillator PUF relies on downstream cryptographic hardware/software to protect the security of the key that is generated. However, there are ways of potentially modifying the ring oscillator PUF's behavior. Such an attack does not reveal the output key, but may be able to influence the device to either fail to regenerate a key (denial of service), or even manipulate secret key bits if such an attack were to occur during provisioning.

For example, it was shown in 2009 that driving a sinusoidal signal on the ground plane of a ring oscillator can cause it to 'lock' to that signal [109]. This study demonstrated such an attack compromising a true random number generator (TRNG). Although an attack on a PUF would have to take into account its differential nature, the same principle can be used. By locking the frequency, an attacker can drive the frequency of a given PUF to a desired value without invasive measures.

In addition, it was shown in 2011, that the electromagnetic radiation from the ring oscillator PUF could also be used to steal the output bits [110]. This attack can be defeated by running several oscillators in parallel, which has been done in many studies on the ring oscillator PUF, some of which predate the identification of the attack [174, 176, 173].

2.4 Shortcomings of Error Correction Techniques

Because POKs are noisy, there must be an error correction phase to enable the production of a stable, secret key. Silicon POK key generation was first introduced using Hamming codes in [62] and more details were presented in [157]. The security argument is information-theoretic. Specifically, if one requires a k-bit secret from n bits generated by the POK, then at most $n - k$ bits could be exposed. The number of correctable errors is quite limited in this approach.

2.4.1 Fuzzy Extractors for Silicon POKs

Fuzzy extractors [54] convert noisy data (e.g., from a silicon POK) into reproducible uniform random strings, which can serve as secret keys in cryptographic primitives to encrypt and authenticate user data.

Fuzzy extractors typically have two phases: a secure sketch (error correction) phase and a privacy amplification (hashing) phase. The secure sketch phase focuses on the recovery of noisy data w. It first outputs a sketch h (also called "helper data") for w. Then, given h and a future measurement w' close to w, it recovers w. The sketch is secure if it does not reveal much about w: w retains much of its entropy even if h is known. This means that h can be stored in public without compromising the privacy of w. However, in typical POK applications, w does not have full entropy, so the privacy amplification phase must compress w prior to obtaining a cryptographic key. In the fuzzy extractor framework, it is possible to extract near-full-entropy keys from a POK source while maintaining information-theoretic security.

The information-theoretic security, however, comes at a high cost in terms of the raw entropy required and the maximum tolerable error rate. The secure sketch phase is well known to lose significant entropy from the helper data h, especially as measurement noise increases. Even in cases where entropy remains after error correction (e.g., [100]), there is not enough entropy remaining to accumulate the 128-bits of entropy in an information-theoretic manner during the privacy amplification phase. According to [87], the entropy loss associated with the use of the information-theoretic entropy accumulator alone is ≥ 128 bits due to the leftover hash lemma.

Works on fuzzy extractors for silicon POKs can be classified based on the additional assumptions they require:

Perfectly i.i.d. Entropy Source. There are several works that created helper data that is information-theoretically secure. [174] uses POK error correction helper data called Index-Based Syndrome (IBS), as an alternative to Dodis' code-offset helper data. IBS is information-theoretically secure, under the assumption that POK output bits are independent and identically distributed (i.i.d.). Given this i.i.d. assumption,

IBS can expose more helper data bits than a standard code-offset fuzzy extractor construction. Efficiency improvements to IBS that maintained information-theoretic security are described in [70] and [71].

Similarly, Paral and Devadas have proposed the use of a "Pattern Matching" technique to correct for errors in candidate strong PUFs for use as a key generation mechanism that also requires an i.i.d. entropy source [123].

A soft-decision POK error correction decoder based on code-offset was described in [98, 99] where the confidence information part of the helper data was proven to be information-theoretically secure under an i.i.d. assumption (the security of the remaining redundancy part associated with the code-offset was not as rigorously addressed in either paper).

Note that while these works created practical implementations based on a provably secure information-theoretic foundation, these works did not explicitly address the full key generation process (secure sketch + privacy amplification); they addressed only the error correction (secure sketch) phase. Further, they need the strong assumption on POK output bits being i.i.d., which allows them to publicly reveal the confidence information. Indeed, POKs are not necessarily i.i.d., and attacks have therefore been performed, e.g., [15]. This approach achieves the same advantage of using confidence information, but it does not reveal this information. Therefore, this proposal remains secure for non-i.i.d. entropy sources (cf. Definition 3.4.1).

Computational Security Based on Machine Learning Heuristics. There were several works [123] [176] [175] that created helper data that is heuristically secure based on results of state-of-the-art machine learning attacks on PUFs [132]. These designs used a candidate strong PUF based on XORs [158] but leak only a limited number of PUF response bits as helper data to generate a key. While several years of attacks by several groups around the world have established the heuristic security of leaking a limited number of bits from a candidate strong PUF [48, 78, 131, 132, 162, 14], there is not yet a proof to reduce this difficulty into a computational hardness assumption accepted by the cryptography community. These works are also limited in scope in that they do not explicitly address the full key generation processing, but

address only the error correction phase.

Secure Sketch + Privacy Amplification. To the best of the author's knowledge, there is one paper that attempted to implement and address the security associated with both stages of a fuzzy extractor [100]. The paper accounted for the information-theoretic loss of the error correction helper data, using code-offset syndrome [54], but did not have sufficient entropy left over from the secure sketch phase to implement an information-theoretically secure privacy amplification stage and instead opted for a more efficient implementation using a lightweight hash called SPONGENT [21] as an entropy accumulator.

Under the assumption that confidence values are independent of the measurement values, information-theoretically secure extractors can also produce a stateless construction as is presented in Section 4.1. However, in this construction, this assumption can be relaxed through a computational hardness assumption of a variant LPN problem.

2.4.2 Computational Fuzzy Extractors

Given the discussion above on information-theoretically secure fuzzy extractors, a more efficient key extraction framework that can be used repeatedly and which is based on an established computational hardness assumption is compelling.

Fuller et al. [58] give a computational fuzzy extractor based on LWE. In Fuller et al.'s scheme, the output entropy improves; the error correction capacity, however, does not. Indeed, Fuller et al. show in their model that secure sketches are subject to the same error correction bounds as information-theoretic extractors. Their construction therefore requires exponential time to correct $\Theta(m)$ errors, where m is the number of bits output by the POK.

The construction presented in Chapter 3 uses Fuller et al.'s LWE construction translated to LPN in the proposed fuzzy extractor. However, this work changes the fuzzy extractor model and leverages the confidence information (common in many POKs) to correct $\Theta(m)$ errors in polynomial time. Fuller et al. expect that it is

unlikely to overcome an exponential complexity in correcting a linear number of errors, since there is no place to securely put a trapdoor in a fuzzy extractor. This work shows that certain kinds of POK sources have dynamically regenerated confidence information that does not require persistent storage memory and can in fact serve as a trapdoor (cf. Sections 3.2 and 3.3). It is also shown that security can be maintained even if the bits generated by the POK are correlated (cf. Definition 3.4.1).

The stateless PUF construction in Section 4.1 requires a restricted version of LPN (cf. Section 3.1.2) with a hardcoded $\mathbf{A}$ matrix. This restriction is important so the construction can repeatedly generate secret keys ($\mathbf{s}$'s) and expose the LPN public keys ($\mathbf{b}$'s) for the same noise ($\mathbf{e}$), while maintaining security based on hardness of LPN and a slight variant of LPN. (This variant LPN assumption is only required for the stateless construction and not the fuzzy extractor scheme.)

2.4.3 Helper Data Manipulation

The issue of helper data manipulation has been addressed with robust fuzzy extractors [26, 53]. Their use of a helper data hash do not address recent helper data manipulation attacks in [82, 49], including ones that take advantage of the linear, bitwise-XOR nature of code-offset helper data as applied to linear error correction codewords.

In the stateless PUF construction in Section 4.1, the helper data comprises (part of) the challenge. Since in LPN-based fuzzy extractor the key is uncorrelated computationally to the helper information, the scheme can authenticate the helper information in a computationally secure manner via a keyed-hash message authentication code such as HMAC [91]. This results in schemes that are secure in a computational sense against active adversaries that modify the helper data in the fuzzy extractor or stateless PUF construction.

3 - LPN Fuzzy Extractor

In this chapter, I introduce a fuzzy extractor construction that leverages LPN and the notion of "confidence" information to efficiently correct errors in bits obtained from silicon POKs. Before describing the construction, I describe the notion of "confidence information", and also introduce the "Learning Parity with Noise" (LPN) problem.

3.1 Background

3.1.1 Confidence Information for Ring Oscillator POK

Following the formalization provided in [10], first define a Physically Obfuscated Key (POK) as a physical function wherein there is only one challenge. A POK returns a single m-bit response, denoted as $\mathbf{e} = \{\mathbf{e}_1, \mathbf{e}_2, \cdots, \mathbf{e}_m\}$ in this work.

If the POK response is completely stable across measurements, then constructing a stable secret key or strong PUF would be trivial: just use the POK output as the secret key. Unfortunately, the POK response in practice will be slightly different each time due to internal noise, i.e.,

$$\mathbf{e} = \mathbf{e}_{\text{const}} + \mathbf{e}_{\text{noise}}$$

where $\mathbf{e}_{\text{const}}$ is the same for each call to the POK, and $\mathbf{e}_{\text{noise}}$ is sampled at random from some distribution over $\{0,1\}^m$. Error-correcting the POK response to tolerate the noise is a major challenge because of tension between several requirements:

1. The POK bits must remain secret (any helper information to error correction must not reveal information about the POK bits).

2. The error-corrected output must be secret and random/pseudorandom and have no noise.
3. One desires to correct as many erroneous POK bits as possible with high probability of success.

The case study will be based on the Ring Oscillator (RO) POK, which generates bits by comparing the frequencies of two ring oscillators that are identical by design, yet whose frequencies vary due to manufacturing variation. Each POK output bit is simply determined by which oscillator is faster. It was first observed in [158] that if the difference in counts between the two ring oscillators is large, then one can have higher confidence that environmental changes are unlikely to cause the output bit to flip erroneously when measured at a later time. This difference will be the confidence information (cf. Figure 3-1). While there have been improvements in ring oscillator structures (e.g., [60]), the case study uses the basic structure of Figure 3-1.

3.1.2 Learning Parity with Noise

The Learning Parity with Noise (LPN) problem is a famous open problem that is widely conjectured to be hard, as the best known algorithm is slightly subexponential ($2^{\Omega(n/\log n)}$) [19, 11]. As a result, this problem has since been used as the foundation of several cryptographic primitives [77, 9, 8, 18].

The problem is posed as follows. Let $\mathbf{s} \in \{0,1\}^n$ be chosen uniformly at random. Let $\mathbf{A} \in \{0,1\}^{m \times n}$ be uniformly random, $m \geq n$. Let $\mathbf{e} \in \{0,1\}^m$ be chosen from a distribution χ. Finally, define $\mathbf{b} \in \{0,1\}^m$ (where $\cdot$ is a dot product) as:

$$\begin{aligned}
\mathbf{b}_1 &= \mathbf{A}_1 \cdot \mathbf{s} + \mathbf{e}_1 \mod 2 \\
\mathbf{b}_2 &= \mathbf{A}_2 \cdot \mathbf{s} + \mathbf{e}_2 \mod 2 \\
\vdots &= \quad \vdots \\
\mathbf{b}_m &= \mathbf{A}_m \cdot \mathbf{s} + \mathbf{e}_m \mod 2
\end{aligned}$$

The problem is to learn $\mathbf{s}$ given only the values of $\mathbf{b}$ and $\mathbf{A}$. When each $\mathbf{e}_i$ is distributed

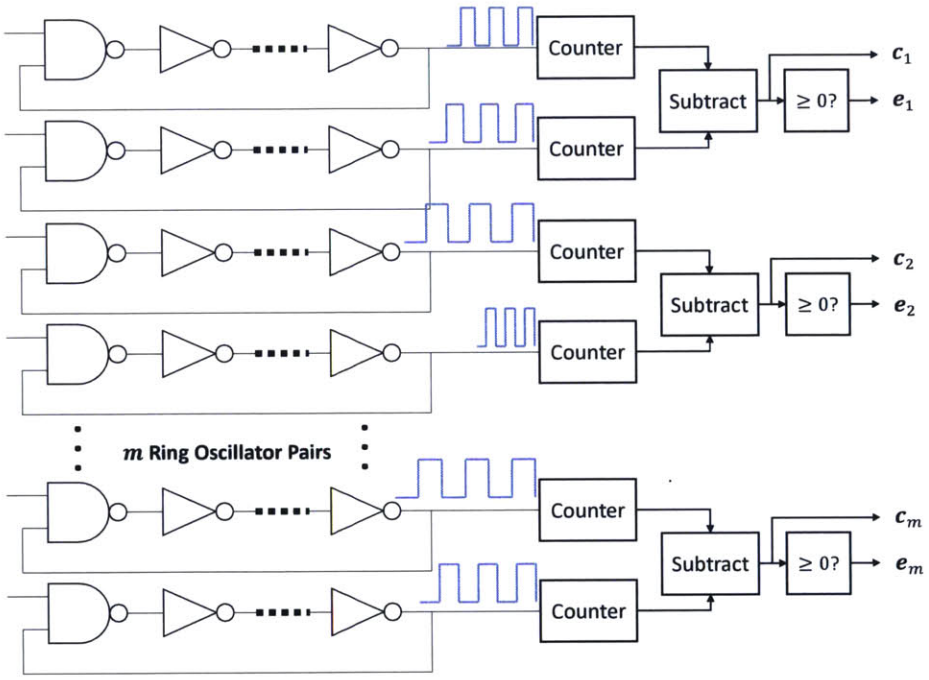


Figure 3-1: A basic Ring Oscillator POK with m differential pairs. Note that in addition to the output bits $\mathbf{e}_i$, confidence values $\mathbf{c}_i$ may be made available to the surrounding logic. These confidence values are in the form of the actual differential count between the two ring oscillators, while the POK output bits $\mathbf{e}_i$ correspond to whether the differential count is greater/less than 0.

according to probability distribution χ.

Conjecture 3.1.1 (LPN Hardness [77]). *There is no algorithm that solves an LPN problem instance* $(\mathbf{A}, \mathbf{b}, \chi)$, *where* $\mathbf{s}$ *and* $\mathbf{A}$ *are uniformly random, in time* $\mathsf{poly}(n, 1/(\frac{1}{2} - \tau))$ *with non-negligible probability in* n, *where* χ *is an Bernoulli distribution with bias* τ.

There is significant evidence to justify the LPN hardness conjecture [126]. The best known algorithms run in $2^{O(n/\log n)}$ [19, 17, 94, 68].

The LPN problem can be thought of as a special case of the Learning With Errors (LWE) problems discussed by Regev [126], by allowing the equations to instead be modulo a prime number q (as opposed to 2). However, Regev's reduction to the shortest independent vector problem (SIVP) does not apply to the LPN case. Therefore, the difficulty of solving LPN is a separate conjecture from the difficulty of solving LWE. This thesis will present a fuzzy extractor and a stateless PUF based on LPN, but the constructions can be extended to the LWE case.

3.2 Fuzzy Extractor Using LPN

This section considers how to reliably reconstruct a key $\mathbf{s}$ from a noisy POK. Start with the fuzzy extractor scheme described in [58], which leverages LWE to extract a pseudorandom string from fuzzy data. First, I translate that work from LWE to LPN discussed in Section 3.1.2.

Construction 3.2.1. *Let* k *be a security parameter, and let* $n = \mathsf{poly}(k)$, *and* $m \geq n$. *Define* $(\mathbf{A}, \mathbf{b}) \leftarrow \mathsf{Gen}(1^k)$, *and* $\mathbf{s} \leftarrow \mathsf{Rep}(\mathbf{A}, \mathbf{b})$ *as follows:*

This construction is exactly analogous to Construction 4.1 of Fuller et al. [58], translated to LPN from LWE (all equations are $\mod 2$ instead of $\mod q$). Therefore, I state the following theorem without proof, as it is analogous to Theorem 4.7 of [58] except under the LPN hardness conjecture.

Theorem 3.2.2. *Let* k *be a security parameter. If Conjecture 3.1.1 is true, then there is a setting of* $n = \mathsf{poly}(k)$ *for which there exists* $\epsilon = \mathsf{neg}(k)$ *such that the following*

1: **procedure** $(\mathbf{A}, \mathbf{b}) \leftarrow \mathsf{Gen}(1^k)$
2: Input $\mathbf{e} \in \{0,1\}^m$ from the POK (modeled by some distribution χ over $\{0,1\}^m$).
3: Sample $\mathbf{A} \in \{0,1\}^{m \times n}$ uniformly at random.
4: Sample $\mathbf{s} \in \{0,1\}^n$ uniformly at random.
5: Compute $\mathbf{b} = \mathbf{A} \cdot \mathbf{s} + \mathbf{e}$.
6: **return** $(\mathbf{A}, \mathbf{b})$.
7: **end procedure**

1: **procedure** $\mathbf{s} \leftarrow \mathsf{Rep}(\mathbf{A}, \mathbf{b})$
2: Input $\mathbf{e}' \in \{0,1\}^m$ from the POK.
3: Let $\mathbf{s} = \mathsf{Decode}_t(\mathbf{A}, \mathbf{b}, \mathbf{e}')$.
4: **return** $\mathbf{s}$.
5: **end procedure**

is true: For any randomized circuit size $s = \mathsf{poly}(k)$ and $t = O(\log n)$ bit errors, Construction 3.2.1 is a $(\{0,1\}^m, \chi, \{0,1\}^{n-o(n)}, t)$ fuzzy extractor that is (ϵ, s)-hard, with failure rate $\delta = e^{-\Omega(k)}$ (cf. Definition 2.5 of [58]).

In the above construction, Decode keeps picking random sets of n equations $\mathbf{b}_i = \mathbf{A}_i\mathbf{s} + \mathbf{e}_i$ and solves for $\mathbf{s}$. If $t = O(\log n)$, Decode succeeds with overwhelming probability after a polynomial number of trials. This thesis will now describe a new extractor algorithm based on LPN that can correct $t = \Theta(m)$ errors in polynomial time. Before presenting the extractor formally, I present an intuitive description.

3.2.1 Intuitive Description

Recall the description of the LPN problem in Section 3.1.2, which is also depicted in Figure 3-2. The above construction uses POK output as the $\mathbf{e}_i$ values. Therefore, an adversary learns the equations with probability of error being $\mathsf{Pr}(\mathbf{e}_i = 1) = \tau$, where τ relates to the entropy of the POK. Having access to the POK allows one to regenerate $\mathbf{e}'_i$, where $\mathsf{Pr}(\mathbf{e}'_i = 1) = \tau$ and $\mathsf{Pr}(\mathbf{e}'_i \neq \mathbf{e}_i) = \tau' \ll \tau$. The regeneration is imperfect due to intrinsic noise of the POK as well as environmental changes. The LPN problem remains hard even for small τ' implying that key recovery will run in exponential time for $\Theta(m)$ number of errors.

A critical enabling property of LPN/LWE is that if one can identify *any* set of

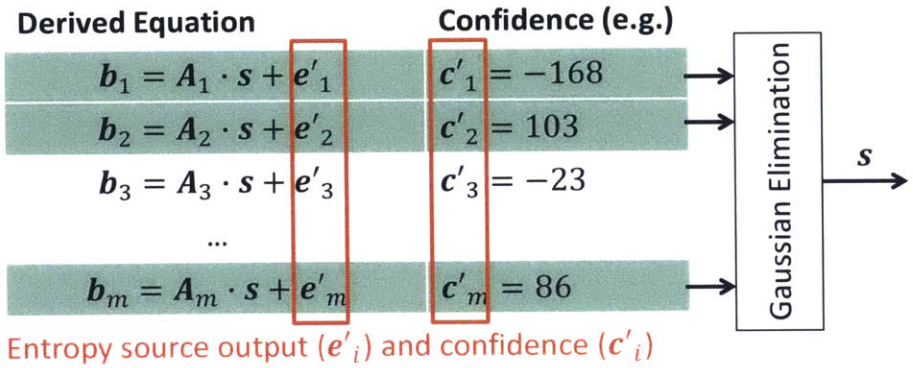


Figure 3-2: Overview of LPN key extraction algorithm. The $\mathbf{e}'_i$ values are regenerated and the $\mathbf{c}'_i$ values with high absolute value identify the $\mathbf{e}'_i$ with low probability of error (since $\mathbf{c}'_i$ values don't change dramatically between measurements, and $\mathbf{e}'_i = \mathsf{Sign}(\mathbf{c}'_i)$). Gaussian elimination is then used on these selected equations to extract the secret key.

n bits that are correct ($\mathbf{e}'_i = \mathbf{e}_i$), then one can use Gaussian elimination to solve for $\mathbf{s}$. Therefore, the key intuition is that access to confidence information during the *regeneration* of the POK bits helps the extractor decide which bits are more likely to be stable (the set of stable bits may be different from measurement to measurement). Then, Gaussian elimination is performed on the set of equations corresponding to these stable bits.

Of course, one must architect the system such that there are enough stable POK bits during each measurement. To this end, the LPN/LWE problem allows *arbitrary redundancy* in the number of equations supplied. Therefore, enough equations can be supplied such that with high probability (see Section 3.3) the recovery succeeds.

Initially, this may sound similar to using the "mask" data in some POK implementations. However, this approach is fundamentally different and has superior security properties, as the POK itself acts as a hidden trapdoor to a hard problem. The confidence information is *discarded* after use and never exposed. The security proof for this new construction will therefore be identical to that of Construction 3.2.1, since the adversary receives identical information.

3.2.2 Detailed Construction

In the following description, I will refer to the POK bits as $\mathbf{e} = \{\mathbf{e}_1, \mathbf{e}_2, \ldots \mathbf{e}_m\}$, ($\mathbf{e}_i \in \{0,1\}$) and their noisy counterparts (measured during response verification) as $\mathbf{e}' = \{\mathbf{e}'_1, \mathbf{e}'_2, \ldots \mathbf{e}'_m\}$, (once again, $\mathbf{e}'_i \in \{0,1\}$). Moreover, the confidence information associated with these noisy measurements will be denoted as $\mathbf{c}' = \{\mathbf{c}'_1, \mathbf{c}'_2 \ldots \mathbf{c}'_m\}$, where $\mathbf{c}'_i \in \mathbb{Z}$ (as shown in Figure 3-2).

The algorithms associated with the key extraction are presented in Algorithm 2. Typically, a fuzzy extractor, information-theoretic or computational, has the functions Gen and Rep, where Gen produces the public helper information, and Rep takes the noisy POK bits and public helper information and returns the error-corrected key. This work expands this construction to include four functions in the extraction process as shown in Algorithm 2:

Before looking into the effects of errors on Algorithm 2, there are several notes to be made.

First, the Fab algorithm can be viewed as system design steps that choose parameters for the desired security and reliability. Project and Recovery together correspond to Rep in a fuzzy extractor.

Second, if the LPN problem is hard, then an adversary in possession of $(\mathbf{A}, \mathbf{b})$ cannot compute $\mathbf{s}$ as shown in Theorem 3.2.2; the adversary obtains no new information. Furthermore, due to the simultaneous hardcore bits of $\mathbf{s}$ in the LPN problem, $\mathbf{s}$ is pseudorandom [7].

Third, the matrix $\mathbf{A}$ can be made a public global system parameter as opposed to per-device output to reduce helper data size; this leaves $\mathbf{b}$ as the only per-device helper data. This will be the same global $\mathbf{A}$ in the stateless PUF construction of Section 4.1, though there exists a much more fundamental security reason to make $\mathbf{A}$ global there.

Lastly, the confidence information $\mathbf{c}'_i$ acts as a trapdoor for identifying "stable" bits in key recovery. Therefore, the key recovery algorithm is faced with a much easier problem and can finish in polynomial time. This will be the focus of the next section.

Algorithm 2 Description of the LPN fuzzy extractor algorithm.

1: **procedure** Fab$(1^k, \delta, \eta)$ // Represents the fabrication step. It takes the security parameter k, the desired probability of recovery failure δ, a η term that characterizes correlation in the POK bits (defined in Definition 3.4.1), and performs the following:
2: Select the size of the secret vector n for the desired security level k based on η (details in Section 4.2).
3: Compute m such that with probability greater than $1 - \epsilon_1$, at least $m' = \Theta(n)$ of the m POK bits are "stable" over relevant noise/environmental parameters. Define "stable" to be $\mathsf{Pr}(\mathbf{e'}_i \neq \mathbf{e}_i) \leq \epsilon_2$. The choice of $m' = \Theta(n)$, values of ϵ_1, ϵ_2 along with other details will be presented in Section 3.3.
4: Manufacture POKs that each produce m bits internally.
5: **end procedure**
6:
7: **procedure** $(\mathbf{A}, \mathbf{b}) \leftarrow$ Gen(n) // Gen takes the size of the secret vector n (calculated in Fab) and returns helper data $\mathbf{b}$:
8: Measure the m POK bits as $\mathbf{e} = \{\mathbf{e}_1, \mathbf{e}_2, \ldots \mathbf{e}_m\}$.
9: Generate a uniformly random secret vector $\mathbf{s} \in \{0,1\}^n$.
10: Compute $\mathbf{b} = \mathbf{A} \cdot \mathbf{s} + \mathbf{e}$.
11: Discard $\mathbf{s}$ and $\mathbf{e}$.
12: Return $\mathbf{b}$.
13: **end procedure**
14:
15: **procedure** $S \leftarrow$ Project$(\mathbf{c'})$ // Represents the algorithm that determines the ''stable'' POK bits'' to be used in Recovery.
16: Use measured confidence information $\mathbf{c'}_i$ to find $m' = \Theta(n)$ stable POK bits.
17: Let S be the set of these stable bits. Return S.
18: **end procedure**
19:
20: **procedure** $\mathbf{s} \leftarrow$ Recovery$(\mathbf{e'}, S)$ // Represents the augmented key recovery algorithm. In addition to the noisy POK measurement $\mathbf{e'}$, this function also takes S, the set of stable bits in $\mathbf{e'}$.
21: Randomly select n out of the m' stable bits.
22: Use Gaussian elimination to solve for $\mathbf{s}$ on the n selected bits.
23: Check if $\mathbf{b}_i = \mathbf{A}_i \cdot \mathbf{s} + \mathbf{e'}_i$ on the remaining $m - n$ equations. An error rate of $\approx 50\%$ implies that the derived $\mathbf{s}$ is incorrect. A significant lower error rate (e.g., 25%) indicates $\mathbf{s}$ is correct.
24: If $\mathbf{s}$ is incorrect, go back to step 1); else output $\mathbf{s}$.
25: **end procedure**

3.3 Noise-Avoiding Trapdoors

In Section 3.2, Project leverages confidence information that a bit is regenerated correctly. This section will explore the asymptotic noise tolerance and efficiency of the system, and the required properties of the POKs to provide confidence information.

$\mathbf{c}_i$ are random variables representing the confidence information of the i'th POK bit at the time of initial challenge-response generation. Next, $\mathbf{c'}_i$ are random variables representing the confidence information of the i'th POK bit at some point in time later. The confidence data are extracted upon measurement of a POK bit, and are never persistently stored.

Define the corresponding POK bit to be a random variable $\mathbf{e}_i = \mathsf{Sign}(\mathbf{c}_i)$, and $\mathbf{e'}_i = \mathsf{Sign}(\mathbf{c'}_i)$. Crucially, if the confidence is high for a particular bit, $\mathsf{Pr}(\mathbf{e'}_i = \mathbf{e}_i) \approx 1$.

To provide concrete analysis, consider the probability distribution of the $\mathbf{c}_i$ and $\mathbf{c'}_i$ random variables, and assume they follow the same zero-mean Gaussian with variance σ_{INTER}, shown in Figure 3-3. Note that this directly implies that $\mathsf{Pr}(\mathbf{e}_i = 1) = \tau = 1/2$ for the LPN problem. In actual physical systems, there will be a bias towards 1 or 0, but it can be seen that assuming a 0.5 bias represents a "worst-case" *from the standpoint of error correction.* (Note that I will use a different worst-case bias for other purposes, e.g., to determine n given the security parameter in Section 4.2.)

Now, given that $\mathbf{c}_i$ and $\mathbf{c'}_i$ represent the random variables for measuring the *same* bit, the conditional distribution $\mathsf{Pr}(\mathbf{c}_i|\mathbf{c'}_i = c)$ is much narrower (where c is the actual measured value of $\mathbf{c'}_i$ at regeneration). This distribution is modeled to be a Gaussian distribution with mean c and variance σ_{INTRA}, also shown in Figure 3-3.

Also note that $\mathbf{c}_i$ and $\mathbf{c'}_i$ represent the same POK bit measured at different times, so they have the same distribution (with no prior knowledge). Therefore, $\forall c$, $\mathsf{Pr}(\mathbf{c}_i|\mathbf{c'}_i = c) = \mathsf{Pr}(\mathbf{c'}_i|\mathbf{c}_i = c)$. In other words, one can use the confidence information collected during the fabrication step to reason about the probability of error at regeneration, or vice-versa.

Now define the probability of error given confidence information. Since $\mathbf{e}_i = \mathsf{Sign}(\mathbf{c}_i)$, the error probability given a measurement of the confidence information is

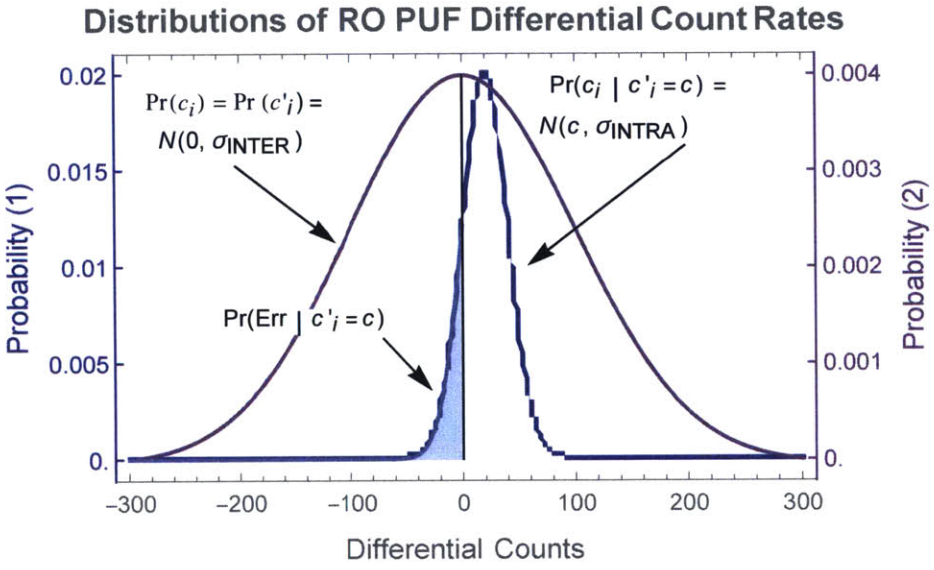


Figure 3-3: Distribution of confidence information for different POK bits when measured repeatedly over time/environmental parameters. The magenta curve corresponds to the distribution of confidence information across different devices. The blue curve corresponds to the distribution of measured confidence information from the same device in different conditions. The probability of error given a confidence measurement c as the integral of the shaded region.

the integral of the shaded region in Figure 3-3, in particular, the CDF up to 0:

$$\mathsf{Pr}(\mathbf{e}'_i \neq \mathbf{e}_i | \mathbf{c}'_i = c) = \frac{1}{2}\left(1 + \mathsf{Erf}\left(-\frac{|c|}{\sqrt{2}\sigma_{\text{INTRA}}}\right)\right) \tag{3.1}$$

3.3.1 Fabrication/Provisioning

Fab must compute m such that with probability greater than $1-\epsilon_1$, at least m' of the m bits will be stable. Recall a random bit $\mathbf{e}_i$ is defined to be stable if $\mathsf{Pr}(\mathbf{e}'_i \neq \mathbf{e}_i) < \epsilon_2$ over relevant environmental parameters.

To do this, recognize that requiring $\mathsf{Pr}(\mathbf{e}'_i \neq \mathbf{e}_i | \mathbf{c}'_i = c) < \epsilon_2$ sets a threshold c_T on $|c|$ in Equation 3.1. If for a particular bit $|c| > c_T$, then the bit is stable. Plug these requirements into Equation 3.1 and solve for c_T (define Erf^{-1} as the inverse of Erf):

$$c_T = \sqrt{2}\sigma_{\text{INTRA}}\mathsf{Erf}^{-1}(1-2\epsilon_2) \tag{3.2}$$

Therefore, the probability that a given bit is stable (has $|c| > c_T$) can be computed by integrating the PDF of $\mathsf{Pr}(\mathbf{c}'_i)$, or equivalently $\mathsf{Pr}(\mathbf{c}_i)$:

$$P_{\text{ST}} = \mathsf{Pr}(|\mathbf{c}_i| > c_T) > 1 - \mathsf{Erf}\left(c_T/(\sqrt{2}\sigma_{\text{INTER}})\right)$$

The inequality is because the probability of a bit being stable is *smallest* when the bit bias is 0.5 (the Gaussian is centered at 0). One can see that as the center of the Gaussian shifts, more probability density falls in the region of $|\mathbf{c}_i| > c_T$. Therefore, the probability of a stable bit can only be higher than the calculation here expects.

Plugging in c_T from Equation 3.2 gives:

$$P_{\text{ST}} > 1 - \mathsf{Erf}\left(\frac{\sigma_{\text{INTRA}}}{\sigma_{\text{INTER}}}\mathsf{Erf}^{-1}(1-2\epsilon_2)\right) \tag{3.3}$$

The final step is to compute m such that at least m' POK bits will be stable with probability $1-\epsilon_1$. This is a binomial distribution and is subject to a Chernoff bound.

Define X as the random variable for the number of stable bits observed.

$$\mathsf{Pr}(X \leq m') \leq \exp\left(-\frac{1}{2}\left(1 - \frac{m'}{mP_{\mathrm{ST}}}\right)^2 mP_{\mathrm{ST}}\right) \leq \epsilon_1$$

Rearranging,

$$m \geq \frac{1}{P_{\mathrm{ST}}}\left(m' - \log(\epsilon_1) + \sqrt{\log(\epsilon_1)\left(\log(\epsilon_1) - 2m'\right)}\right) \tag{3.4}$$

Since P_{ST} is a function of ϵ_2, given m', ϵ_1, and ϵ_2, one can compute m such that at least m' of the POK bits are stable with probability $1 - \epsilon_1$, as is required.

3.3.2 Projection/Extraction and Showing the "Trapdoor"

The extension of the above analysis to the Project algorithm is comparatively simple. Project simply selects $m' = \Theta(n)$ bits that have measured confidence $\mathbf{c'}_i = c$ where $|c| > c_T$. Because of the Fab algorithm, one can be confident that m' bits are stable with overwhelming probability.

The stable bits above only guarantee $\mathsf{Pr}(\mathbf{e'}_i \neq \mathbf{e}_i | \mathbf{c'}_i = c) < \epsilon_2$, but "truly stable" bits (bits without error) are required perform Gaussian elimination. Define t' as the number of bits that are not truly stable. If $\epsilon_2 = \Theta(1/n)$, then $\mathsf{E}(t') = \Theta(1)$, and a Chernoff bound shows that:

$$\mathsf{Pr}\left(t' > \alpha \log n\right) < e^{-\Theta(\log^2 n)}.$$

So $t' = o(\log n)$ with overwhelming probability.

Recovery randomly selects n out of the m' bits to perform Gaussian elimination. If the n selected are all "truly stable," Gaussian elimination on them will yield the

correct $\mathbf{s}$ and Recovery succeeds. The above probability is given by

$$\frac{\binom{m'-t'}{n}}{\binom{m'}{n}} > \left(1 - \frac{t'}{m'-n}\right)^n \approx \exp\left(-\frac{nt'}{m'-n}\right) = \frac{1}{\mathsf{poly}(n)}$$

Therefore, after $\mathsf{poly}(n)$ number of iterations, Recovery finds the correct $\mathbf{s}$ with overwhelming probability. The overall failure probability–accounting for all types of failures (have less than m' stable bits, $t' = \omega(\log n)$, or fail to select n truly stable bits in all iterations)–is at most $\epsilon_1 + (1-\epsilon_1)\mathsf{negl}(n)$. Set $\epsilon_1 = \Theta(2^{-n})$ to get overall negligible failure probability.

Without the confidence trapdoor, the LPN hardness states exactly that it is infeasible to compute $\mathbf{s}$ in polynomial time with non-negligible success probability. Therefore, while an adversary requires exponential time to calculate $\mathbf{s}$, the owner of the fuzzy extractor requires only polynomial time. This is the definition of a trapdoor.

3.3.3 Setting m

Set $\epsilon_1 = \Theta(2^{-n})$, $\epsilon_2 = \Theta(1/n)$ and $m' = \Theta(n)$. To compute m from Equation 3.3 and 3.4, the ratio $\sigma_{\text{INTRA}}/\sigma_{\text{INTER}}$ must be characterized as it determines P_{ST}.

Define $\sigma_r = \sigma_{\text{INTRA}}/\sigma_{\text{INTER}}$. Consider a *worst-case*: $\sigma_r = 1$. In this case, Equation 3.3 reduces to $P_{\text{ST}} = 2\epsilon_2$. Plug them into Equation 3.4, and one obtains:

$$m = \frac{1}{2\epsilon_2}\left(m' + n + \sqrt{n(n+2m')}\right) = \Theta\left(n^2\right)$$

In reality, the ratio $\sigma_r < 1$, so the hidden constant in $\Theta(n^2)$ is small, as will be seen in Section 3.5.

3.3.4 Improving on the Trapdoor

The above asymptotic result will be improved if one assumes $\sigma_r = o(1)$. For example, pick σ_r such that P_{ST} is asymptotically constant.[1]

To accomplish this, recognize that $\mathsf{Erf}^{-1}(1-2\epsilon_2) \leq \sqrt{-\log(\epsilon_2)}$ as $\epsilon_2 \to 0$ [39]. Therefore, if $\sigma_r \leq c/\sqrt{-\log(\epsilon_2)}$ for some constant c, then $P_{\mathrm{ST}} \geq 1 - \mathsf{Erf}(c)$ for all n, and therefore $m = \Theta(n)$.

Note that the bound of $\sigma_r \leq c/\sqrt{-\log(\epsilon_2)}$ is very close to constant. For example, set $c = 1$ and $\epsilon_2 = 1/n$. For $n = 128, 256$, I find $\sigma_r < 0.45, 0.42$ respectively.

Finally, consider the effect of $\sigma_r \leq c/\sqrt{-\log(\epsilon_2)}$ on the number of correctable errors of the fuzzy extractor. Integration of the conditional probability distribution in Equation 3.1 (cf. Figure 3-3) results in the associated marginal distribution (the error probability):

$$\mathsf{Pr}(\mathbf{e}'_i \neq \mathbf{e}_i) = \frac{1}{2} - \frac{1}{\pi}\tan^{-1}(1/\sigma_r)$$

A constant σ_r clearly implies $\Theta(m)$ errors.

For $\sigma_r = o(1)$, $\mathsf{Pr}(\mathbf{e}'_i \neq \mathbf{e}_i) = \Theta(\sigma_r)$ as $\sigma_r \to 0$. This implies that with $m = \Theta(n)$ one can no longer correct $\Theta(m)$ errors asymptotically; instead, the maximum number of correctable errors is $O(m\sigma_r)$. For practical key sizes the impact on error correction is minimal.

3.4 LPN Fuzzy Extractor Security Analysis and Assumptions

The proof of security for an LPN fuzzy extractor using confidence information is identical to Theorem 3.2.2. This is because the additional confidence information (which may or may not be correlated with the actual value of the POK bit) described in Section 3.3 that is used to help extract the key is never revealed.

[1] Note that one can make $\epsilon_2 = \Theta(\log(n)/n)$ and still brute-force correct in polynomial time. This does not impact the asymptotic analysis later in this section, so I ignore it.

3.4.1 Assumptions on POK Outputs

The POK outputs are used as the noise term in the LPN problem, and the construction is secure if the POK outputs are i.i.d. I now provide a significantly relaxed definition of POK source entropy under which the fuzzy extractor construction remains secure. In particular, the following definition describes the class of sources that are secure with this construction.

Definition 3.4.1. *Define a set of L different m-bit entropy sources whose probability distribution may be constructed in the following way:*

1. *Begin with a set X of $m \times L$ bits that are i.i.d. with $\mathsf{Pr}(X_i = 1) = \eta$, $1 > \eta > 0$.*
2. *Select a set of affine linear transformations $F = \{F_0, F_1, \ldots, F_k\}$ (where $F(X) = M \cdot X + N$ for some $mL \times mL$ matrix M, and mL-dimensional vector N). Select a k-bit string f according to an arbitrary[2] distribution over $\{0,1\}^k$.*
3. *Return $F_k^{f_k}(F_{k-1}^{f_{k-1}}(\cdots F_1^{f_1}(F_0^{f_0}(X))\cdots))$, where $F_i^1 = F_i$ and F_i^0 is the identity transformation.*

This distribution is clearly much more general than an i.i.d. distribution, as it allows for certain bits from the same/different entropy sources to be correlated. For example, consider the i'th bit of LPN problem A, and the j'th bit of LPN problem B, this distribution can support a non-zero correlation coefficient between these bits, namely, $\mathsf{Corr}(\mathbf{e}_{A,i}, \mathbf{e}_{B,j})$. However, it is tighter than min-entropy, as min-entropy allows for individual bits to be "stuck" at one or zero. In this distribution, bits cannot be perfectly correlated (e.g., $\mathsf{Corr}(\mathbf{e}_{A,i}, \mathbf{e}_{B,j}) = 1$). η in effect sets the "maximum correlation", and η (as well as $1 - \eta$) must not be 0 or negligible in the security parameter.

Note also that knowledge of which bits are correlated is *public* (it is assumed that the adversary knows the transformations that are applied). Furthermore, note that the set of bits X is the set of bits *across* different sources. For this discussion, each source has m bits. If there are L different sources, then X is the set of all $L \times m$ bits. As a result, correlations between bits on different sources is allowable in the

[2]that can be sampled in polynomial time

definition. Under this assumption, Lemma 3.4.2 proves the security of the system.

Lemma 3.4.2. *If the entropy sources for a collection of LPN fuzzy extractors have a joint distribution that can be described by Definition 3.4.1 for some η, then an algorithm that can extract* $\mathbf{s}$ *from any of the fuzzy extractors in polynomial time with non-negligible advantage can be used to solve the traditional LPN problem with bias η in polynomial time with non-negligible advantage.*

Lemma 3.4.2 can be proved by recognizing that a set of LPN problems with i.i.d. bits for their e_i values can be converted into a collection of LPN problems with bits described by Definition 3.4.1 by probabilistically applying the identified sequence of linear transformations F to their public keys $(\mathbf{A}, \mathbf{b})$. The proof is given:

Proof. Consider a collection of L different m-bit entropy sources. Let X be the set of all $m \times L$ bits, and let the joint distribution of X be described by Definition 3.4.1. Specifically, Definition 3.4.1 takes several parameters. Let the initial bias be η. Let $F = \{F_0, F_1, \ldots, F_k, \ldots\}$ be the set of affine transformations. Let $P = \{P_0, P_1, \ldots, P_k, \ldots\}$ be the set of random bits that determines which subset of F_i are applied. Let P have some joint distribution. The definition states that one can sample from this distribution in polynomial time.

Now, consider the set of L corresponding LPN problems (each using a distinct set of m bits from X as its e_i values). Let adversary $\mathcal{A}$ take as argument the public parameters of this set of LPN problems: $(\mathbf{A}_i^j, \mathbf{b}_i^j)$, for i from 1 to m (there are m equations in a single LPN problem), and j from 1 to L (the set of L LPN problems). Assume that there exist parameters η, F, and a distribution over P such that $\mathcal{A}$ calculates at least one of the secret keys of the set of LPN problems with non-negligible probability.

Using $\mathcal{A}$, construct algorithm $\mathcal{B}$ that takes as argument the public parameters of L different LPN problems whose $\mathbf{e}_i$ bits are i.i.d. with bias η. $\mathcal{B}$ will return the secret vector of at least one of the LPN problems with non-negligible probability. Note that $\mathcal{B}$ is equivalent to breaking the LPN problem, as each LPN problem is independent.

$$
\begin{array}{l}
\text{Problem 1}: \left\{ \begin{array}{l} \mathbf{b}_1^1 = \mathbf{A}_1^1 \cdot \mathbf{s}^1 + \mathbf{e}_1^1 \\ \mathbf{b}_2^1 = \mathbf{A}_2^1 \cdot \mathbf{s}^1 + \mathbf{e}_2^1 \\ \vdots \\ \mathbf{b}_m^1 = \mathbf{A}_m^1 \cdot \mathbf{s}^1 + \mathbf{e}_m^1 \end{array} \right. \\
\text{Problem 2}: \left\{ \begin{array}{l} \mathbf{b}_1^2 = \mathbf{A}_1^2 \cdot \mathbf{s}^2 + \mathbf{e}_1^2 \\ \mathbf{b}_2^2 = \mathbf{A}_2^2 \cdot \mathbf{s}^2 + \mathbf{e}_2^2 \\ \vdots \\ \mathbf{b}_m^2 = \mathbf{A}_m^2 \cdot \mathbf{s}^2 + \mathbf{e}_m^2 \end{array} \right. \\
\vdots \\
L \text{ Problems}
\end{array}
\tag{3.5}
$$

First, consider a single LPN problem where $\mathbf{A} = \{\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_m\}$, $\mathbf{b} = \{\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_m\}$, $\mathbf{e} = \{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_m\}$, and $\mathbf{b}_i = \mathbf{A}_i \cdot \mathbf{s} + \mathbf{e}_i$. The bits $\mathbf{e}_i$ have some distribution. The key recognition is that the act of applying an affine transformation to the set of bits $\mathbf{e}$ is equivalent to applying the same transformation to $\mathbf{A}$ and $\mathbf{b}$. If one wants to transform the distribution by applying $F(e) = M \cdot e + N$ (M is an $m \times m$ dimensional matrix and N is an m-dimensional vector), then one can derive a different LPN problem:

$$
\begin{aligned}
F(\mathbf{b})_i &= F(\mathbf{A} \cdot \mathbf{s} + \mathbf{e})_i \\
(M \cdot \mathbf{b} + N)_i &= (M \cdot \mathbf{A})_i \cdot \mathbf{s} + F(\mathbf{e})_i
\end{aligned}
$$

By setting $\mathbf{b}' = M \cdot \mathbf{b} + N$ and $\mathbf{A}' = M \cdot \mathbf{A}$, a new LPN problem is constructed: $\mathbf{b}'_i = \mathbf{A}'_i \cdot \mathbf{s} + \mathbf{e}'_i$, where $\mathbf{e}'_i = F(\mathbf{e})_i$. By modifying *only the public parameters*, an affine transformation is performed on the distribution of $\mathbf{e}_i$.

This may be generalized to multiple LPN problems by recognizing that the above technique can be applied to the set of equations that comprise multiple LPN problems by simply concatenating the vectors, resulting in Equation 3.5.

Now, recognize (where | is concatenation) that to transform a set of problems with $\mathbf{e}^1|\mathbf{e}^2|\cdots\mathbf{e}^L$ into a set of problems with $F(\mathbf{e}^1|\mathbf{e}^2|\cdots\mathbf{e}^L)$, one can simply concatenate the aforementioned observation (note that M is now an $mL \times mL$ sized matrix, and

N is a vector of dimension mL):

$$\mathbf{b'}^1|\mathbf{b'}^2|\cdots\mathbf{b'}^L = M\cdot(\mathbf{b}^1|\mathbf{b}^2|\cdots\mathbf{b}^L)+N$$
$$\mathbf{A'}^1|\mathbf{A'}^2|\cdots\mathbf{A'}^L = M\cdot(\mathbf{A}^1|\mathbf{A}^2|\cdots\mathbf{A}^L)$$

I now return to the discussion of algorithm $\mathcal{B}$. The algorithm $\mathcal{B}$ is the probabilistic application of the above fact multiple times. The steps of $\mathcal{B}$ are as follows:

1. Sample p_i from the distribution of each P_i.
2. Set $\mathbf{b}_{\text{TOT}} = \mathbf{b}^1|\mathbf{b}^2|\cdots\mathbf{b}^L$.
3. Set $\mathbf{A}_{\text{TOT}} = \mathbf{A}^1|\mathbf{A}^2|\cdots\mathbf{A}^L$.
4. For j from 0 to k, define $F_j(x) = M_j\cdot x+N_j$. If $p_j = 1$, set $\mathbf{b}_{\text{TOT}} = M_j\cdot\mathbf{b}_{\text{TOT}}+N$, and set $\mathbf{A}_{\text{TOT}} = M\cdot\mathbf{A}_{\text{TOT}}$. Otherwise, do nothing.
5. Call $\mathcal{A}$ using the newly created public parameters for the set of LPN problems. Return the secret vector that $\mathcal{A}$ computes.

The final value of the public parameters corresponds to a set of LPN problems where the statistics of $\mathbf{e}_i^j$ are equal to those that can be solved by $\mathcal{A}$. Moreover, the set of secret vectors has not changed, and this problem was obtained by modifying bits of the public parameters *only*. Therefore, if $\mathcal{A}$ exists, it will recover at least one $\mathbf{s}$ with non-negligible advantage. Then, $\mathcal{B}$ can be used to break an i.i.d. LPN problem with bias η. This is a contradiction if LPN is hard, so $\mathcal{A}$ cannot exist. □

Note that the key step in the above algorithm is that $\mathcal{B}$ applies the affine transformation to the public parameters of the set of LPN problems. This operation produces a new set of LPN problems that are *statistically identical* to the case where the POK bits themselves have a transformed distribution. This allows $\mathcal{B}$ to transform a set of i.i.d. LPN problems into a set of LPN problems with correlated POK data by modifying only public parameters, and without affecting the secret vectors.

Also note that a corollary of Lemma 3.4.2 is that $\mathbf{s}$ remains pseudorandom even in the presence of correlated bits. This is due to the fact that LPN's secret has $n - o(n)$ simultaneous hardcore bits [7], and is proven for uncorrelated LWE in [58]. The proof is similar for the correlated LPN construction, as it is independent of the

transformations performed in Lemma 3.4.2.

3.4.2 Security Parameter Derivation

The security goal is that an adversary given helper data must perform $\Omega(2^k)$ operations (k is the security parameter) to discover the secret key. I show below that a key size of $n = 128$ results in a security parameter of $k = 128$ against the best known attacks. The equality of key size and security parameter is unusual for security constructions with formal hardness reduction, and is especially unusual for LPN cryptosystems.

There are two key factors enabling this property. First, recognize that typical LPN-based cryptosystems must have a low error rate (e.g., $\tau = \Pr(\mathbf{e}_i = 1) = 0.0024$ [44]) to ensure correct decryption/verification. This construction, on the other hand, does not use any LPN encryption/decryption algorithm, and therefore *does not have the same restriction on* τ. In fact, τ can theoretically be set to 0.5, representing full entropy in the POK data. However, real POK data is not ideal and may not have full entropy. To be conservative, pessimistically assume $\tau = \eta = 0.4$ and that the POK bits are correlated in a way that LPN is still hard (formalized in Definition 3.4.1 and Lemma 3.4.2).

The second factor is that number of equations in the construction is limited to $m \in O(n^2)$. Current best LPN algorithms are based on the BKW algorithm [19], which requires $m = 2^{O(n/\log n)}$. In order to successfully attack the LPN fuzzy extractor, one would have to use the technique from Lyubashevsky [97], which works with $m = O(n^{1+\epsilon})$ equations but immediately increases the runtime to $2^{O(n/\log\log n)}$.

The idea of Lyubashevsky's algorithm is to generate more equations from the given $m = O(n^{1+\epsilon})$ equations, increasing the noise rate to

$$\tau_L = \frac{1}{2} - \frac{1}{2}\left(\frac{1-2\tau}{4}\right)^{\frac{2n}{\epsilon \log n}} \tag{3.6}$$

and then using other LPN algorithms, such as BKW [19], LF1, LF2 [94] as a black box with the increased error rate. For $m = \Theta(n^2)$ ($\epsilon = 1$), $n = 128$ and $\tau = 0.4$,

	Time Complexity	Security Parameter
BKW	$na\left(2^{b+1}(1-2\tau_L)^{-2^a}\ln\left(\frac{b}{\theta}\right)+(a-1)2^b\right)$	2^{247}
LF1	$b2^b+na\left(8\ln\left(\frac{2^b}{\theta}\right)(1-2\tau_L)^{-2^a}+(a-1)2^b\right)$	2^{135}
LF2	$3\cdot 2^b na+b2^b$	N/A

Table 3.1: Comparison of performance of LPN algorithms against an LPN fuzzy extractor with $\tau_L = \frac{1}{2} - 1.31 \times 10^{-48}$, $n = 128$. Set $\theta = 1/3$ to achieve 50% success probability [22]. The security parameter is taken for optimal choices of a, b (not shown). The security parameter of LF2 is N/A, because there is no setting of parameters that results in the algorithm converging.

$\tau_L = \frac{1}{2} - 1.31 \times 10^{-48}$.

The recent analysis from [22] shows that the LF1, LF2 algorithms empirically have the best performance in the limit of high noise ($\tau_L \to 0.5$). Table 3.1 compares BKW, LF1, LF2. Note that each of the above algorithms performs worse than brute-force or does not succeed at all. Therefore, $n = 128$ for a security parameter of $k = 128$.

3.5 Case Study using a Ring Oscillator POK

I will use Ring Oscillator POKs as a case study because of the easy availability of confidence information (cf. Figure 3-1). In the case of the RO POK, the differential counts between the ring oscillators is the confidence information $\mathbf{c}'_i$, and the output bit $\mathbf{e}'_i = \mathsf{Sign}(\mathbf{c}'_i)$ described in Section 3.1.1.

I have provided a theory explaining the resilience of the LPN construction to noise and environmental parameters using this confidence information in Section 3.3. Now, this theory and collected data from a set of 320 pairs of ring oscillators measured across temperature and voltage ranges is used to demonstrate the efficiency of the LPN fuzzy extractor construction in a concrete fashion. Experiments were conducted on a Xilinx Virtex 7 Series Field Programmable Gate Array (FPGA).

Differential counts were measured for a set of 320 ring oscillator pairs in a wide (beyond industrial) range of temperature and voltage. Three interesting points are -40°C@$0.95V$, 25°C@$1.00V$, and 105°C@$1.05V$. Other ranges that are used are the differential count values at commercial (0°C to 70°C) and extended industrial (-40°C

Temp.	Bias
$-40°C$	54%
25°C	52%
105°C	53%

Temp.	σ_{INTRA}
$-40°$C	24.3 ± 1.3
0°C	8.9 ± 0.40
70°C	17.4 ± 0.64
85°C	24.0 ± 1.0
105°C	33.7 ± 1.4

Table 3.2: (Left) Measured bias of 320 RO pairs at varying temperatures. (Right) Measured σ_{INTRA} for varying temperatures.

to 85°C). The $\sigma_{\text{INTRA}}/\sigma_{\text{INTER}}$ ratios improve as the temperature range is reduced.

Note that 24% of the ring oscillator pairs produce different responses in the environmental range; this is the typical $O(m)$ error case for such circuits under environmental stresses in the ranges shown.

The measured bias of the RO counts across temperature is shown in Table 3.2. Therefore, the pessimistic estimate of bias ignoring correlation effects as 45% (or 55% equivalently) is correct.

These differential count values are distributed according to the distribution discussed in Section 3.3 with variance σ^2_{INTER}. For each of these temperatures the distribution of differential counts was Gaussian, as was assumed in Section 3.3. Each of the fits from which the parameters are derived have a reduced $\chi^2 \approx 1$, indicating that the Gaussian model is a good fit to the data within experimental error. Moreover, neither the mean nor variance of the distribution changed significantly over temperature or voltage. Therefore, one can describe the distribution in terms of a single mean, variance (μ_{INTER}, σ_{INTER}) shown in Figure 3-4. To measure μ_{INTRA} and σ_{INTRA}, one must measure the distribution of how these differential counts change regardless of the differential count measured at provisioning. This distribution is $\mathsf{Pr}(\mathbf{c'}_i - \mathbf{c}_i)$. This distribution was calculated by using data from different ring oscillators. The standard deviation of this distribution is σ_{INTRA}.

Room temperature is used as a baseline (this would be the condition in which the challenge-response pairs would be initially generated). Next, I measured how the differential counts change as temperature/voltage vary for each of the 320 ring oscillator pairs. These data provide a statistical distribution of how much the differential

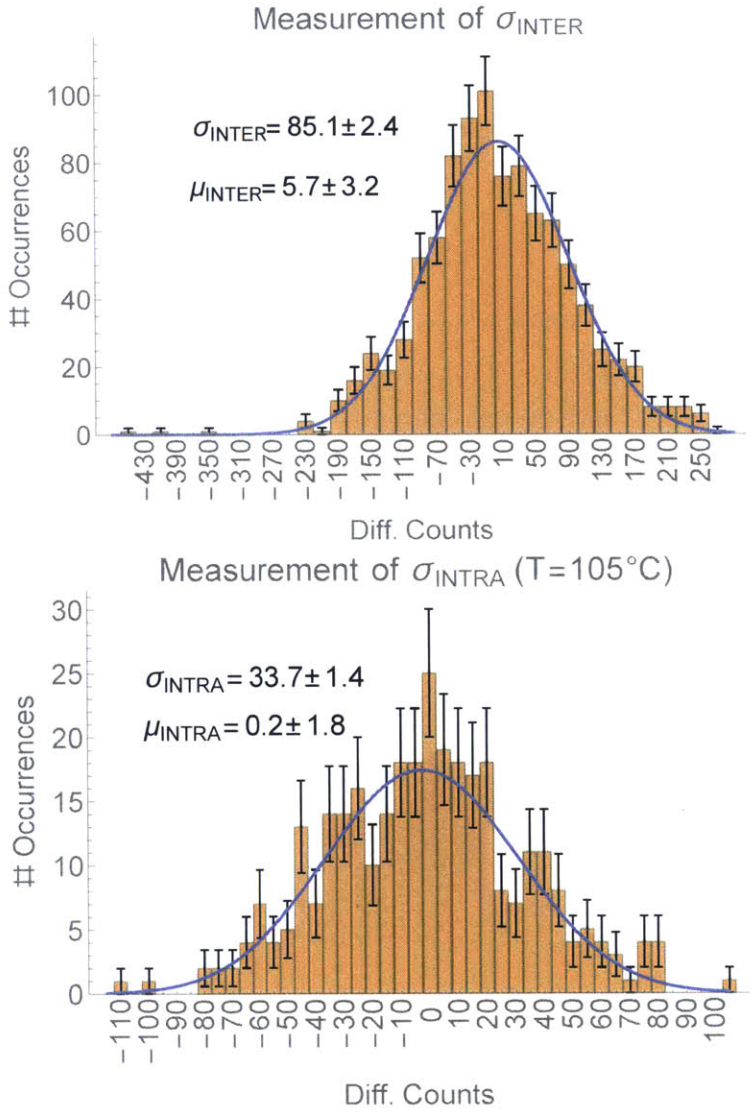


Figure 3-4: (Top) Measurement of σ_{INTER} through the estimation of the distribution of differential counts across 320 RO pairs across room temperature and the fast and slow voltage/temperature corners. (Bottom) Measurement of σ_{INTRA} by subtracting differential counts at 25°C@1V from 105°C@1.05V.

count value will change with a change in environmental parameters (the distribution described by σ_{INTRA}, μ_{INTRA} in Section 3.3).

The distribution at 105°C is shown in Figure 3-4. The measurements at various temperatures are shown in Table 3.2. It is important to note that although in Section 3.3 no theoretical justification was presented for the reason why the distribution of counts of a single ring oscillator pair over relevant environmental conditions would be Gaussian, this does turn out to be the case within experimental error as demonstrated in Figure 3-4. Using these measurements, I calculate the ratio $\frac{\sigma_{\text{INTRA}}}{\sigma_{\text{INTER}}}$ for commercial (0°C to 70°C) as 0.20, extended industrial (−40°C to 85°C) as 0.29, and the maximum temperature range the experiment could support (−40°C to 105°C) as 0.40. This is summarized in Table 3.3.

I now present an analysis of the resource requirements (number of RO pairs) of the LPN fuzzy extractor scheme with a security parameter of 128, and probability of error 10^{-6} over the above temperature ranges.

First, the theoretical construction in Section 3.2 is far too conservative for practical purposes. In practice, simply choose the most stable $m' = n$ bits, and most likely there are at most $t' \leq 1$ error bits in them. For example, if $\epsilon_2 = 3 \times 10^{-6}$, a simple binomial distribution analysis shows that $\mathsf{Pr}(t' > 1) < 10^{-6}$. Therefore, an exhaustive search over the error bit with a Gaussian elimination operations for each will suffice.

Plugging $\epsilon_1 = 10^{-6}$, $\epsilon_2 = 3 \times 10^{-6}$, $m' = n = 128$ (giving a security parameter of 128) and $\frac{\sigma_{\text{INTRA}}}{\sigma_{\text{INTER}}}$ values into Equations 3.3, 3.4, I compute m (the total number of RO pairs) for various temperature ranges, also shown in Table 3.3.

Note that the extracted bitstring is *not pseudorandom* with security parameter 128. In order to obtain a pseudorandom bitstring for use as a key, one must either use SHA-1 or similar (a random oracle). To avoid the use of such a function, one may also double the LPN secret size to $n = 256$ and then select an arbitrary subset of 128 bits. These 128 bits would be pseudorandom by the result from [7].

Note that the analysis is still pessimistic (e.g., assuming that all stable bits have error probability ϵ_2 even though most bits have much lower error probability) and the construction is unoptimized. Even with an unoptimized implementation, these results

Temp.	Erroneous Bits	$\frac{\sigma_{\text{INTRA}}}{\sigma_{\text{INTER}}}$	# Ring Osc. Pairs ($= m$)
0°C – 70°C	9%	0.20	450
−40°C – 85°C	21%	0.29	770
−40°C – 105°C	24%	0.40	1870

Table 3.3: Summary of $\frac{\sigma_{\text{INTRA}}}{\sigma_{\text{INTER}}}$ and resources required for an LPN fuzzy extractor over the specified temperature range. The percentage of erroneous bits over environmental conditions and associated ratio is displayed. Extraction succeeds with error probability $< 10^{-6}$ and a security parameter of 128.

compare very well with the works described in Section 2.4. For example, PUFKY [100] requires 2052 helper data bits for a 10°C – 80°C temperature range, compared to 450 helper data bits for a comparable 0°C – 70°C temperature range, and 770 helper data bits for a much wider temperature range -40°C – 85°C. Moreover, unlike most prior work on information theoretic extractors, the LPN fuzzy extractor can be scaled to higher noise settings simply by increasing m without affecting the security argument.

4 - LPN Stateless PUF

Now, I extend the LPN fuzzy extractor from Chapter 3 to be a fully "stateless" strong PUF. The fuzzy extractor from the previous chapter extracts a single key, while the construction in this section allows for *multiple challenge response pairs* to be generated from a single set of noisy bits (i.e., a ring oscillator POK).

This construction is developed according to the formalism provided in [10], and the security reduction shows that the LPN stateless PUF is "strongly unpredictable" (cf. Definition 4.1.2, which is analogous to the same concept from [10]) if LPN is hard.[1] These reductions are in the random oracle model.

4.1 Stateless PUF Construction

4.1.1 Stateless PUF Definition

A *Stateless PUF* is a pair of functions $\mathbf{PUF} = \{\mathsf{Gen}_{\mathrm{POK}}, \mathsf{Ver}_{\mathrm{POK}}\}$ with access to a POK, where $\mathsf{Gen}_{\mathrm{POK}}$ is responsible for generating and outputting challenge-response pairs, while $\mathsf{Ver}_{\mathrm{POK}}$ takes a challenge as input, and outputs a response. The intent is for $\mathsf{Gen}_{\mathrm{POK}}$ to be called multiple times by a verifier over a secure channel to obtain a collection of challenge/response pairs. At a later time, the verifier will send one of these challenges to the PUF over an insecure channel, to which the PUF must generate the correct response. A challenge-response pair therefore can only be used once by $\mathsf{Ver}_{\mathrm{POK}}$.

[1] If the PUF noise is i.i.d., then the reduction is directly to LPN, otherwise the reduction is to a variant of LPN (cf. Conjecture 4.2.3).

Definition 4.1.1. *A (m, l, χ) stateless PUF is a pair of randomized probabilistic polynomial time procedures $\{\mathbf{c}, \mathbf{r}\} \leftarrow \mathsf{Gen}_{\mathrm{POK}}(1^k)$, and $\mathbf{r} \leftarrow \mathsf{Ver}_{\mathrm{POK}}(\mathbf{c})$ where*

- *The challenge-response generation algorithm $\mathsf{Gen}_{\mathrm{POK}}(1^k)$ takes as argument the security parameter k. It returns a challenge-response pair $\{\mathbf{c}, \mathbf{r}\}$, with $\mathbf{c}, \mathbf{r} \in \{0,1\}^*$ and $|\mathbf{c}|, |\mathbf{r}| \in \mathsf{poly}(k)$. The subscript* POK *corresponds to the POK contained within the PUF. That is, each PUF manufactured will have a unique POK according to distribution χ over $\{0,1\}^m$ due to manufacturing variation.*
- *The verification algorithm $\mathbf{r} \leftarrow \mathsf{Ver}_{\mathrm{POK}}(\mathbf{c})$ takes as input a challenge $\mathbf{c}$, and returns the corresponding response $\mathbf{r}$. Again,* POK *refers to the unique POK contained within the PUF.*

Now define the security of the Stateless PUF (s − uprd refers to "strong unpredictability" as defined in [10]).

Definition 4.1.2 (Stateless PUF Strong Security)**.** *A stateless PUF is ϵ-secure with error δ if $\mathsf{Pr}\left[\{\mathbf{c}, \mathbf{r}\} \leftarrow \mathsf{Gen}_{\mathrm{POK}}(1^k) : \mathbf{r} = \mathsf{Ver}_{\mathrm{POK}}(\mathbf{c})\right] > 1 - \delta$ and for all PPT A, $\mathsf{Adv}^{\mathrm{s-uprd}}_{\mathbf{PUF}}(\mathsf{A}) < \epsilon$, which is defined in terms of the following experiment.*

1: **procedure** $\mathsf{Exp}^{\mathrm{s-uprd}}_{\mathbf{PUF}}(\mathsf{A})$
2: Make polynomial queries to $\mathsf{Gen}_{\mathrm{POK}}(\cdot), \mathsf{Ver}_{\mathrm{POK}}(\cdot)$
3: if A returns $\{\mathbf{r}, \mathbf{c}\}$ such that:
- $\mathsf{Gen}_{\mathrm{POK}}$ did not return $\{\mathbf{r}, \mathbf{c}\}$.
- $\mathsf{Ver}_{\mathrm{POK}}(\mathbf{c}) = \mathbf{r}$.

4: **then return** 1
5: **else return** 0.
6: **end procedure**

The s − uprd *advantage of* A *is defined as*

$$\mathsf{Adv}^{\mathrm{s-uprd}}_{\mathbf{PUF}}(\mathsf{A}) = \mathsf{Pr}\left[\mathsf{Exp}^{\mathrm{s-uprd}}_{\mathbf{PUF}}(\mathsf{A}) = 1\right] \tag{4.1}$$

While other formalizations of PUF system security have been proposed [10], ours is slightly different in that in the above case, there is *no distinction* between helper data and challenge data. Moreover, the PUF is responsible for generating both the challenge and the response for the verifier to use later.

One key recognition in the above definition is that there is *no provisioning stage.* The algorithms $\mathsf{Gen}_{\mathsf{POK}}$ and $\mathsf{Ver}_{\mathsf{POK}}$ may be called in arbitrary order as many times as required. Put differently, there is no stage at which a secret is programmed into the device or an irreversible operation is performed on the device. This is critical, as the overall system can therefore be stateless, and not have to have any additional protections against adversaries attempting to break the provisioning logic of the device.

The formalism of manufacturing unclonability remains the same as that put forth in [10].

4.1.2 The Construction

I provide concrete constructions for $\mathsf{Gen}_{\mathsf{POK}}$ and $\mathsf{Ver}_{\mathsf{POK}}$ below, which are also illustrated in Figure 4-1.

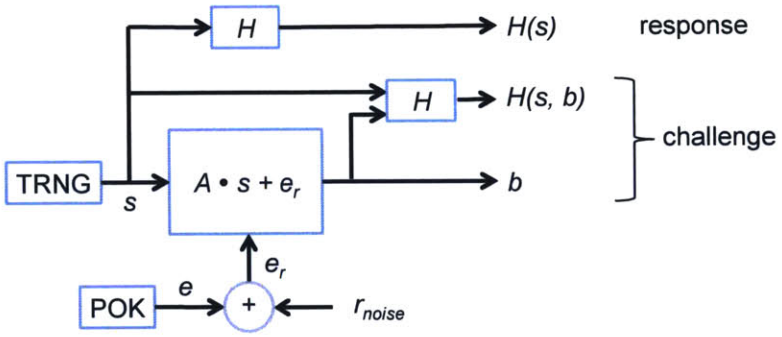
Construction 4.1.3 (LPN Stateless PUF). *Let k be a security parameter, with $m, n \in \mathsf{poly}(k)$, and $m > n$. Define $\mathbf{A} \in \{0,1\}^{m \times n}$ be a uniformly random but* **constant** *and publicly known matrix row-indexed by i from 1 to m. Let both algorithms have access to the random oracle $H(\cdot)$.*

---

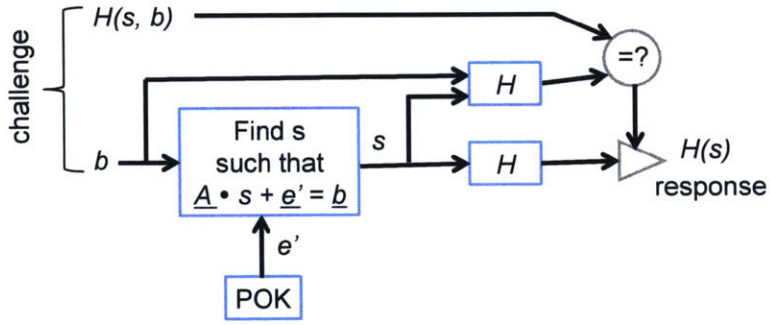
procedure $\{\{\mathbf{b}, \mathbf{D_b}\}, \mathbf{D_s}\} \leftarrow \mathsf{Gen}_{\mathsf{POK}}(1^k)$
  Generate $\mathbf{s} \in \{0,1\}^n$ uniformly at random.
  Regenerate $\mathbf{e} \in \{0,1\}^m$ from POK.
  Compute $\mathbf{b} = \mathbf{A} \cdot \mathbf{s} + \mathbf{e}$.
  return $\{\{\mathbf{b}, H(\mathbf{s}, \mathbf{b})\}, H(\mathbf{s})\}$.
end procedure

procedure $\mathbf{D_s} \leftarrow \mathsf{Ver}_{\mathsf{POK}}(\{\mathbf{b}, \mathbf{D_b}\})$
  Regenerate $\mathbf{e} \in \{0,1\}^m$ from POK.
  Extract $\mathbf{s}$ from $\mathbf{b}$ (details in Section 3.2).
  Verify that $\mathbf{D_b} = H(\mathbf{s}, \mathbf{b})$, else return $\perp$.
  return $H(\mathbf{s})$
end procedure

Note that the above construction requires both internal randomness as well as a random oracle.



(a) $\mathsf{Gen}_{\mathrm{POK}}$: Generation of challenge-response pairs. TRNG stands for True Random Number Generator. r_{noise} is random noise injected into low-confidence bits in the variant construction ($\mathsf{Gen\_Noisy}_{\mathrm{POK}}$) and is 0 for the basic construction ($\mathsf{Gen}_{\mathrm{POK}}$).



(b) $\mathsf{Ver}_{\mathrm{POK}}$: Regeneration of response when the PUF is presented with a valid challenge. The underlines on $\mathbf{A}$, $\mathbf{e}'$ and $\mathbf{b}$ indicate that a subset of n of the m rows are selected to solve for $\mathbf{s}$.

Figure 4-1: Stateless PUF construction. Note that $\mathsf{Gen}_{\mathrm{POK}}$ and $\mathsf{Ver}_{\mathrm{POK}}$ can be called any number of times in any order. The PUF does not retain any state across invocations.

4.1.3 Remarks

Blocking Malicious Challenges

The binding $H(\mathbf{s}, \mathbf{b})$ is included in the challenge-response generation, and $\mathsf{Ver}_{\mathrm{POK}}$ checks if $\mathbf{D_b} = H(\mathbf{s}, \mathbf{b})$ before returning a response. This is important, as Definition 4.1.2 allows for active adversaries. Without this check, an attacker can trivially win the security experiment by returning an output $\mathbf{b}$ by $\mathsf{Gen}_{\mathrm{POK}}$ with one bit modified; the modified bit is likely not used in the recovery of $\mathbf{s}$ at all, and $\mathsf{Ver}_{\mathrm{POK}}$ will accept, trivially violating strong unpredictability. With this check, if $\mathbf{s}$ is recovered correctly,

any modification to $\mathbf{b}$ will be detected with overwhelming probability.

Hash Function Requirements

$H(\cdot)$ is a random oracle that is well-approximated by the SHA-256 or SHA-3 hash functions, which is denoted $H'(\cdot)$. The construction requires H' to be one-way, since $H'(\mathbf{s})$ is exposed. To ensure that an adversary cannot impersonate a PUF, one must require non-malleability of H'. That is, the adversary should not be able to generate $H'(\mathbf{s_1} + \Delta\mathbf{s})$ given $H'(\mathbf{s_1})$ and $\Delta\mathbf{s}$. These properties are required because of the use of $H(\mathbf{s}, \mathbf{b})$ in the construction.

Controlled PUF

I have described a "vanilla" scheme for authentication where responses are returned in the clear when challenges are applied. However, all the controlled PUF (CPUF) protocols of [63] with small modifications are enabled by the construction. Briefly, the verifier obtains a *single* challenge-response pair securely, i.e., no eavesdroppers, as before. When the PUF receives a challenge, it does not return the response, but merely generates it internally and bit-exactly. Now, the verifier who knows the response, can use it as a shared secret for repeated nonce-based authentication or secure communication. Other verifiers can use completely different shared secrets.

4.2 Stateless PUF Security Analysis and Assumptions

In the Stateless PUF construction, $\mathsf{Gen}_{\mathrm{POK}}$ is run multiple times with roughly the same noise term $\mathbf{e} = \mathbf{e}_{\mathrm{const}} + \mathbf{e}_{\mathrm{noise}}$. This deviates from the LPN problem, where the noise term for each equation is required to be independent. Therefore, I will need additional assumptions. For the construction to reduce to LPN, one must assume that the confidence information (i.e., bias of $\mathbf{e}_{\mathrm{noise}}$) is independent of the *actual measurement* of the constant component $\mathbf{e}_{\mathrm{const}}$. Alternatively, the construction can be

reduced to a new conjecture I call Partial-Error-Reuse LPN (PER_LPN), which says LPN is hard even when part of the error bits are reused.

4.2.1 Reduction to LPN Assuming Independence Between Confidence and $\mathbf{e}_{\text{const}}$

Start by noting that in order for the construction to reduce to LPN, $\mathsf{Gen}_{\mathsf{POK}}$ must use the same matrix $\mathbf{A}$ on every query to it. Otherwise, an adversary receives two sets of equations with the same $\mathbf{e}$ (I do not want to rely on the small noise $\mathbf{e}_{\text{noise}}$ in POK output for security),

$$\mathbf{b} = \mathbf{A} \cdot \mathbf{s} + \mathbf{e} \mod 2$$
$$\mathbf{b}' = \mathbf{A}' \cdot \mathbf{s}' + \mathbf{e} \mod 2$$

The adversary can add up the equations mod 2, thereby canceling out the $\mathbf{e}$ terms, and trivially recovers both $\mathbf{s}$ and $\mathbf{s}'$. However, Lemma 4.2.2 will show that if the $\mathbf{A}$ matrix is the same for the different secrets, discovering any individual secret requires breaking standard LPN. Intuitively, this means access to $\mathsf{Gen}_{\mathsf{POK}}$ does not help the adversary.

Next, Lemma 4.2.1 shows that access to $\mathsf{Ver}_{\mathsf{POK}}$ does not help an adversary.

Lemma 4.2.1. *Given an adversary* A *that has non-negligible* $\mathsf{Adv}^{\mathrm{s-uprd}}_{\mathrm{PUF}}(\mathsf{A})$*, there exists an algorithm* B *that makes no queries to* $\mathsf{Ver}_{\mathsf{POK}}$ *and still has non-negligible* $\mathsf{Adv}^{\mathrm{s-uprd}}_{\mathrm{PUF}}(\mathsf{B})$.

Proof. Let algorithm B run A, simulating calls to $\mathsf{Gen}_{\mathsf{POK}}$, $\mathsf{Ver}_{\mathsf{POK}}$ with the following $\mathsf{Gen}_{\mathsf{B,POK}}$ and $\mathsf{Ver}_{\mathsf{B,POK}}$: Responses of $\mathsf{Gen}_{\mathsf{POK}}$ are faithfully relayed to A after being recorded. Queries to $\mathsf{Ver}_{\mathsf{POK}}$ are simulated by always returning $\perp$ (unless the query is made with an output of $\mathsf{Gen}_{\mathsf{B,POK}}$, in which case the recorded value is returned).[2]

By definition, A generates with non-negligible probability a query for which $\mathsf{Ver}_{\mathsf{POK}}$ would *not* return $\perp$. Therefore, A *can distinguish* $\mathsf{Ver}_{\mathsf{B,POK}}$ from $\mathsf{Ver}_{\mathsf{POK}}$. However,

[2]Note that $\mathsf{Ver}_{\mathsf{B,POK}}$ is indistinguishable from $\mathsf{Ver}_{\mathsf{POK}}$ if A has negligible advantage in guessing $\mathbf{s}$.

```
1: procedure {{b, D_b}, D_s} ← Gen_{B,POK}(1^k)
2:     Run {{b, D_b}, D_s} ← Gen_POK(1^k)
3:     Store {{b, D_b}, D_s} to table T.
4:     return {{b, D_b}, D_s}
5: end procedure
6: procedure D_s ← Ver_{B,POK}({b, D_b})
7:     if {b, D_b} ∈ T then return D_s
8:     else return ⊥
9:     end if
10: end procedure
```

regardless of this fact, A must always emit at least one query to $\mathsf{Ver}_{\mathsf{B,POK}}$ for which $\mathsf{Ver}_{\mathsf{POK}}$ would not return $\perp$.[3]

Given that A makes at most a polynomial number of queries to $\mathsf{Ver}_{\mathsf{B,POK}}$, B may choose any of the queries made by A to $\mathsf{Ver}_{\mathsf{POK}}$ at random and have a non-negligible advantage of returning the "correct" query that would be accepted by $\mathsf{Ver}_{\mathsf{POK}}$. Therefore, B has non-negligible $\mathsf{Adv}^{\mathrm{s-uprd}}_{\mathrm{PUF}}(\mathsf{B})$. □

Now, Lemma 4.2.2 presents the security reduction to LPN.

Lemma 4.2.2. *Let k be a security parameter, $n = \mathsf{poly}(k)$, and $m \geq n$. If Conjecture 3.1.1 is true, there is no PPT* A *that has advantage* $\mathsf{Adv}^{\mathrm{s-uprd}}_{\mathbf{PUF}}(\mathsf{A})$ *non-negligible in k.*

Proof. Assume that a PPT algorithm A has non-negligible advantage in the experiment in Definition 4.1.2. According to Lemma 4.2.1, there exists a PPT algorithm B that has non-negligible advantage in the experiment without making queries to $\mathsf{Ver}_{\mathsf{POK}}$. Using B, construct an algorithm C that violates the hardness Conjecture 3.1.1.

Algorithm C takes as input a random LPN problem $(\mathbf{b}, \mathbf{A})$, where $\mathbf{A} \in \{0,1\}^{m \times n}$, $\mathbf{b} \in \{0,1\}^m$, and $\mathbf{b} = \mathbf{A} \cdot \mathbf{s} + \mathbf{e}$, where $\mathbf{s} \in \{0,1\}^n$ is uniformly random, and $\mathbf{e}$ is chosen according to distribution χ. While in standard LPN, χ represents an i.i.d. distribution of m bits, the reduction here also applies to the correlated LPN in Lemma 3.4.2.

[3] After this query, A may have "distinguished" that it is querying $\mathsf{Ver}_{\mathsf{B,POK}}$ instead of $\mathsf{Ver}_{\mathsf{POK}}$, so the behavior of A is undefined.

When B makes calls $H(\cdot)$, C faithfully returns the output of $H(\cdot)$, but records all queries to and responses from $H(\cdot)$. When B makes calls to $\mathsf{Gen}_{\mathsf{POK}}$, C responds using the following simulated version $\mathsf{Gen}_{\mathsf{C,POK}}$:

---

procedure $\{\{\mathbf{b}', \mathbf{D}'_{\mathbf{b}}\}, \mathbf{D}'_{\mathbf{s}}\} \leftarrow \mathsf{Gen}_{\mathsf{C,POK}}(1^k)$
  Generate uniformly random $\Delta\mathbf{s}$.
  $\mathbf{b}' = \mathbf{b} + \mathbf{A} \cdot \Delta\mathbf{s} + \mathbf{e}_{\text{noise}} = \mathbf{A}(\mathbf{s} + \Delta\mathbf{s}) + \mathbf{e} + \mathbf{e}_{\text{noise}}$
  Uniformly generate $U_1,\ U_2 \in \{0,1\}^l$.
  Insert $\{\{\mathbf{b}', U_1\}, U_2\}$ into a local table T.
  return $\{\{\mathbf{b}', U_1\}, U_2\}$.
end procedure

The output $\mathbf{b}'$ by $\mathsf{Gen}_{\mathsf{C,POK}}$ corresponds to the LPN problem with the random secret $(\mathbf{s} + \Delta\mathbf{s})$, and is indistinguishable from the output of $\mathsf{Gen}_{\mathsf{POK}}$. Note that the added $\mathbf{e}_{\text{noise}}$ models the noisy POK output.[4] Assume the distribution of $\mathbf{e}_{\text{noise}}$ does not depend on the constant component of the noise term ($\mathbf{e}$ here), so C can sample the confidence information from $N(0, \sigma_{\mathsf{INTER}})$, and then sample $\mathbf{e}_{\text{noise}}$ from $N(c, \sigma_{\mathsf{INTRA}})$ on its own according to the distributions in Figure 3-3. Note that this implies that the POK noise is i.i.d.

Furthermore, since $H(\cdot)$ is a random oracle, the output U_1, U_2 by $\mathsf{Gen}_{\mathsf{C,POK}}$ are computationally indistinguishable from $\mathbf{D}_{\mathbf{b}}, \mathbf{D}_{\mathbf{s}}$ by $\mathsf{Gen}_{\mathsf{POK}}$. Therefore, C precisely mimics the behavior of $\mathsf{Gen}_{\mathsf{POK}}$ for B, and with non-negligible probability, B outputs $\{\{\mathbf{b}', \mathbf{D}'_{\mathbf{b}}\}, \mathbf{D}'_{\mathbf{s}}\}$ that is not in table T and makes $\mathsf{Ver}_{\mathsf{POK}}$ accept.

$$\mathbf{b}' = \mathbf{A} \cdot \mathbf{s}' + \mathbf{e}'$$
$$\mathbf{D}'_{\mathbf{b}} = H(\mathbf{s}', \mathbf{b}')$$
$$\mathbf{D}'_{\mathbf{s}} = H(\mathbf{s}')$$

Since $H(\cdot)$ is a random oracle, B must have queried $H(\cdot)$ with s' before; otherwise, the probability of $\mathbf{D}'_{\mathbf{s}} = H(\mathbf{s}')$ must be negligible. C has recorded all the queries to and responses from $H(\cdot)$, and thus can retrieve $\mathbf{s}'$ and compute $\mathbf{e}'$.

[4] A POK with i.i.d. noise (assumed in this reduction) is modeled by a constant set of bits ($\mathbf{e}$ in the algorithm) plus some i.i.d. "noise" ($\mathbf{e}_{\text{noise}}$ in the algorithm) with some Bernoulli parameter τ. Therefore, the summation of $\mathbf{e} + \mathbf{e}_{\text{noise}}$ accurately models if the Bernoulli parameter of $\mathbf{e}_{\text{noise}}$ is τ.

In order for $\mathsf{Ver}_{\mathrm{POK}}$ to accept B's output, $\mathbf{e}'$ must be distributed according to the confidence information C sampled. C can then recover $\mathbf{e}$ in the same way Recovery does, and then solve for $\mathbf{s}$. If $\mathsf{Ver}_{\mathrm{POK}}$ accepts B's output with non-negligible probability, then C recovers $\mathbf{e}$ and $\mathbf{s}$ with non-negligible probability. This contradicts Conjecture 3.1.1. □

4.2.2 Reduction to PER_LPN

The above security proof requires that χ is i.i.d., a strong assumption. This is because the reduction requires that the $\mathbf{e}_{\mathrm{noise}}$ be independent of $\mathbf{e}$. However, one may consider the impact of using a more relaxed distribution.

Consider a joint distribution χ_S for m bits, where a subset $S \subset [m]$ of size cn are always 0 (c is a constant) and the remaining $m - cn$ bits are i.i.d. These cn bits are "confident" bits, and are always 0 (there is perfect correlation between the confidence of a bit and its value). The subset U of the other $m - cn$ bits are non-"confident." In this case, the above reduction cannot hold, since $\mathbf{e}_{\mathrm{noise}}$ is not independent of $\mathbf{e}$.

However, use the intuition from Section 3.3 and let $\mathsf{Gen}_{\mathrm{POK}}$, $\mathsf{Ver}_{\mathrm{POK}}$ be able to detect a bit's confidence information. Specifically, let $\mathsf{Gen}_{\mathrm{POK}}$ and $\mathsf{Ver}_{\mathrm{POK}}$ on each query to POK receive $\mathbf{e}$ and also receive an arbitrary, potentially random subset $T \subset S$ of size n. Let it also receive the list U of the $m - cn$ non-"confident" bits.

The above information models confidence information in a pessimistic way. In the model described in Section 3.3, confident bits are approximately noiseless. The other bits have different levels of noise. In the above simplified model, all "stable bits" are noiseless, and everything else is chosen i.i.d.

With this knowledge, $\mathsf{Gen}_{\mathrm{POK}}$ may deliberately introduce i.i.d. noise into the non-confident bits to make the LPN problem hard without affecting the runtime of $\mathsf{Ver}_{\mathrm{POK}}$. This is illustrated in Figure 4-1a using r_{noise}. Consider $\mathsf{Gen\_Noisy}_{\mathrm{POK}}$, and $\mathsf{Ver\_Noisy}_{\mathrm{POK}}$ below. As discussed above, I modify POK to also return T, U, where $T \subset S$ is an arbitrary, potentially random subset of the stable bits S. U is the set of $m - cn$ unstable bits. For technical reasons pertaining to the reduction below, I require that Gen_Noisy and Ver_Noisy generate/verify a polynomial number

of $\{\{\mathbf{b}, H(\mathbf{s},\mathbf{b})\}, H(\mathbf{s})\}$. The reason for this will become apparent in the reduction.

---

procedure $\{\{\mathbf{b}, \mathbf{D_b}\}, \mathbf{D_s}\} \leftarrow \mathsf{Gen\_Noisy}_{\mathrm{POK}}(1^k)$
  repeat
    Generate $\mathbf{s} \in \{0,1\}^n$ uniformly at random.
    Query $\{T, U, \mathbf{e}\} \leftarrow$ POK
    Set $\mathbf{e}_i$ to uniform random $\{0,1\}$ for all $i \in U$.
    Compute $\mathbf{b} = \mathbf{A} \cdot \mathbf{s} + \mathbf{e}$.
    Store $\{\{\mathbf{b}, H(\mathbf{s},\mathbf{b})\}, H(\mathbf{s})\}$ into Tab_1.
  until Polynomial L Iterations
  return Tab_1
end procedure

procedure $\mathbf{D_s} \leftarrow \mathsf{Ver\_Noisy}_{\mathrm{POK}}(\mathrm{Tab}_2)$
  Set $L = \mathrm{Length}(\mathrm{Tab}_2)$.
  Initialize $\mathrm{ErrSum} = 0^m$.
  for each $\{\mathbf{b}, \mathbf{D_b}\}$ in Tab_2 **do**
    Query $\{T, U, \mathbf{e}\} \leftarrow$ POK
    Run Gaussian elimination on $T \subset [m]$ equations to compute $\mathbf{s}$ (details in Section 3.2).
    Verify that $\mathbf{D_b} = H(\mathbf{s},\mathbf{b})$, else return $\perp$.
    Add $H(\mathbf{s})$ to Tab_3
    $\mathbf{e_{meas}} = A \cdot s - b \mod 2$
    $\mathrm{ErrSum} = \mathrm{ErrSum} + \mathbf{e_{meas}}$
  end for
  Verify for all $i \in U$ that $\mathrm{ErrSum}_i \approx L/2$, else return $\perp$.
  return Tab_3
end procedure

The overall bias of the LPN equation returned by Gen_Noisy is therefore $(m - cn)/(2m)$, and $m - cn$ of the equations have i.i.d. noise. If $cn \ll m$, then this bias is very close to $\frac{1}{2}$ and most of the bits are i.i.d. Moreover, no adversary can identify which of the m bits are stable. Therefore, the security of a cryptosystem with the above statistics reduces to the following slightly modified LPN conjecture, which I term "Partial Error Reuse LPN", or PER_LPN. It is parameterized by c, m.

Conjecture 4.2.3 ($\mathsf{PER\_LPN}_{c,m}$). *Consider a distribution χ where LPN is hard (e.g., χ may be Bernoulli and Conjecture 3.1.1 holds) for m samples and a key of size n. LPN is also hard for $\mathbf{e}$ distributed as χ_S, where $S \subset [m]$ with cn elements is secret. χ_S is constructed as follows:*

1. *Sample a set of m bits $\mathbf{e}$ with distribution χ.*

2. *Set* $\mathbf{e}_i = 0$ *for all* $i \in S$.

I am now ready to show that $\mathbf{PUF\_Noisy} = \{\mathsf{Gen\_Noisy}, \mathsf{Ver\_Noisy}\}$ comprises a Stateless PUF according to Definition 4.1.1. First, recognize that Lemma 4.2.1 may still be applied. Therefore, one only must provide A with polynomial samples of $\{\{\mathbf{b}, \mathbf{D_b}\}, \mathbf{D_s}\}$ from $\mathsf{Gen\_Noisy}$.

Lemma 4.2.4. *Let* k *be a security parameter,* $n = \mathsf{poly}(k)$*, and* $m \geq n$*. If Conjecture 4.2.3 is true, there is no PPT* A *that has advantage* $\mathsf{Adv}^{\mathsf{s-uprd}}_{\mathbf{PUF\_Noisy}}(\mathsf{A})$ *non-negligible in* k*.*

Proof. Given A, construct B that takes a set of $\mathsf{PER\_LPN}$ problems as an argument (**PerLPNTab**) and returns the secret parameters of *all* members of this set.

1: **procedure** $\mathbf{s_{tab}} \leftarrow \mathsf{B}$(**PerLPNTab**)
2: Initialize ErrSum $= 0^m$.
3: Generate χ' such that $\mathsf{Pr}(\chi_S + \chi')$ is equivalent to the noise distribution of $\mathsf{Gen\_Noisy}$.
4: **for** each $\mathbf{b} \in$ **PerLPNTab do**
5: Sample $\mathbf{e_{new}}$ from χ'.
6: Set $\mathbf{b} = \mathbf{b} + \mathbf{e_{new}}$.
7: **end for**
8: `// Use A to recover new queries.`
9: Record queries by A to $H(\cdot)$ into **HTab**.
10: Call **NewPerLPNTab** $\leftarrow \mathsf{A}$(**PerLPNTab**)
11: `// Compute Distribution of Errors returned by A.`
12: **for** each $\mathbf{b}' \in$ **NewPerLPNTab do**
13: Find $\mathbf{s}'$ in **HTab** s.t. $\mathsf{Ham}(\mathbf{b}' + \mathbf{A} \cdot \mathbf{s}') \not\approx m/2$.
14: Compute $\mathbf{e}' = \mathbf{b}' - \mathbf{A} \cdot \mathbf{s}' \mod 2$
15: Set ErrSum $=$ ErrSum $+ \mathbf{e}'$.
16: **end for**
17: `// Identify ''Confident'' Bits`
18: Select $S \subset [m]$ where $\mathrm{ErrSum}_i \not\approx L/2$ for all $i \in S$.
19: `// Solve original PER_LPN problems.`
20: Pick a random subset $T \subset S$ of size n.
21: Perform Gaussian elimination on this subset of equations for all $\mathsf{PER\_LPN}$ instances to recover each $\mathbf{s}$. Add each to $\mathbf{s_{tab}}$
22: **return** $\mathbf{s_{tab}}$.
23: **end procedure**

Algorithm B uses the $\mathsf{PER\_LPN}$ error distribution and a separate distribution χ' to emulate the distribution of the POK (Lines 3-7 of B).

B then queries A with these samples, while recording calls to $H(\cdot)$. Using the returned values from A and the recorded queries from $H(\cdot)$, B can construct a polynomial-sized set of pairs $(\mathbf{b}', \mathbf{s}')$. By the Definition of A, these $(\mathbf{b}', \mathbf{s}')$ pairs correspond to a challenge that will make Ver_Noisy accept with non-negligible probability.

Next, B uses the returned values from A to recover the secret subset $S \subset [m]$. That is, it recovers *which bits are "confident"* and do not change. B can do this by looking at the distribution of errors returned by A. Steps 11-18 of B perform this analysis.

In essence, these steps use the set of L pairs $(\mathbf{b}', \mathbf{s}')$ to compute $\mathsf{Pr}(\mathbf{e}'_i) = 1$ for $i \in [m]$. If $\mathsf{Pr}(\mathbf{e}'_i) \not\approx 1/2$, then it is likely that $i \in S$.

Note that this explains why the game was modified earlier (in Gen_Noisy and Ver_Noisy) to incorporate a polynomial number of challenge/response pairs. A larger number of random samples are needed to accurately characterize the noise distribution.

Moreover, it is required that A correctly samples from this distribution in Ver_Noisy by ensuring that elements of U are uniform random (but *only* for positions in U).

Therefore, since A makes Ver_Noisy accept with non-negligible probability, it must set $\mathsf{Pr}(\mathbf{e}_i = 1/2)$ for all bits $i \in U$.

Therefore, B can use these samples constructed by A to distinguish between U and S. Note that Algorithm B requires that χ' be bitwise distinguishable from uniform random.

Now that B has recovered the "confident" bits, it may select a random subset of these bits and use Gaussian elimination to solve for $\mathbf{s}$ in each of the PER_LPN instances. This contradicts Conjecture 4.2.3. □

Lemma Remarks: Lemma 4.2.4 requires that the POK distribution be constructed from $\mathsf{Pr}(\chi_S + \chi')$, where χ' does not depend on the output of χ_S. This is achievable only if the distribution of a confident bit does not depend on which other bits are confident. E.g., the following distribution is not allowed: $\mathsf{Pr}(\mathbf{e}_1 = 1) = 0$ if $\mathbf{e}_1, \mathbf{e}_2$ are both confident, but $\mathsf{Pr}(\mathbf{e}_1 = 1) = 1$ if $\mathbf{e}_1$ is confident, but $\mathbf{e}_2$ is *not* confident.

Second, Lemma 4.2.4 requires that the distribution over the confident bits χ' be

bitwise distinguishable from the uniform distribution ($\mathsf{Ber}_{1/2}$). Algorithm B computes $\mathsf{Pr}(\mathbf{e}'_i = 1)$ where either $\mathbf{e}'_i \sim \chi'$ or $\mathbf{e}'_i \sim \mathsf{Ber}_{1/2}$. Algorithm B must be able to distinguish each bit sampled from χ' from $\mathsf{Ber}_{1/2}$. In practice, this corresponds to χ' having the property that the marginal probability of each bit equaling '1' be measurably different from $1/2$.

Next, there is a slight abuse of notation in lines 15 of $\mathsf{Ver\_Noisy}$, 13 and 18 of B through the use of $\approx$ and $\not\approx$. In each of these steps, the algorithm is given a set of bits and asked to determine if they have Hamming weight close to some value. The purpose is to determine whether the bits have bias $1/2$.

Finally, it is no longer required that the matrix $\mathbf{A}$ be *constant* for each challenge/response pair. Informally, this is because $\mathsf{Gen\_Noisy}_{\mathrm{POK}}$ deliberately introduces noise into $\mathbf{e}$, making subsequent challenge/response pairs difficult to combine. Furthermore, Conjecture 4.2.3 is strengthened to allow only some of the bits in $\mathbf{e}$ to be random. Formally, recognize that the proof for Lemma 4.2.4 does not require $\mathbf{A}$ to be constant. Note that maintaining constant $\mathbf{A}$ does not decrease security, so practical implementations may continue to use constant $\mathbf{A}$.

4.2.3 Stateless PUF Theorem

The security theorem for the stateless PUF construction can now be stated. If the POK bits are i.i.d. (which are used as the noise term $\mathbf{e}$), the theorem holds under Conjecture 3.1.1. For more complex distributions of POK bits, use Conjecture 4.2.3.

Theorem 4.2.5. *Let k be a security parameter, and $n = \mathsf{poly}(k)$. There exists a choice of n, $m \geq n$, $l \geq k$, χ such that Construction 4.1.3 is a (m, l, χ) stateless PUF that is ϵ-secure with error δ, with $\epsilon = \mathsf{neg}(k)$, $\delta = \mathsf{neg}(k)$ under either Conjecture 3.1.1 or Conjecture 4.2.3 depending on χ.*

Proof. First, recognize that Construction 4.1.3 is efficient. Clearly, $\mathsf{Gen}_{\mathrm{POK}}$ runs in polynomial time. Section 3.3.2 shows $\mathsf{Ver}_{\mathrm{POK}}$ runs in polynomial time. Second, under either Conjecture 3.1.1 or Conjecture 4.2.3, there does not exist any PPT A that gains has advantage $\mathsf{Adv}^{\mathrm{s-uprd}}_{\mathbf{PUF}}(\mathsf{A}) > \epsilon$, where $\epsilon = \mathsf{neg}(k)$. □

5 - Introduction to Public Model Physical Unclonable Functions

In a PUF, instead of using secret data stored in a traditional non-volatile digital memory, the device being authenticated uses a unique data based on physical characteristics of the IC that cannot be duplicated. In the case of silicon PUFs, this unique fingerprint results from manufacturing variability in the integrated circuit [64]. This approach has a number of advantages: (1) Simple hardware design, (2) threat surface for extraction attempts limited to "power-on" state, and (3) does not use expensive non-volatile memory.

However, there are also several significant drawbacks to this authentication approach. First, a client authenticating a PUF on a server cannot compute challenge-response pairs. Therefore, it must first establish a secure channel, collect a table of random challenge-response pairs, and then re-use these pairs to authenticate the server over a public channel at a later time. The client must securely store the challenge-response table and each challenge-response pair can only be used once. Therefore, the PUF can only be challenged a certain number of times before the client must collect more challenge-response pairs from the server (requiring the secure channel to be re-established).

PPUFs leverage many of the concepts of traditional PUFs, but take these ideas several steps further. A PPUF system consists of physical "PPUF hardware", and a "PPUF model" that may be hardware, software, or a hybrid of both. Intuitively, a PPUF operates in the uses the same challenge-response approach as a standard PUF with two major differences. First, instead of using a secure bootstrap to create a

secret list of challenge-response pairs, the PPUF hardware has an associated PPUF model. This model is a program or device that can compute the response for any challenge. Second, this model is *public.*

Just as in the PUF model, there can only be one PPUF hardware due to unclonability. In the PPUF system however, the associated model can be freely copied and distributed. This leads to the second requirement: there must be a measurable difference in the time it takes the PPUF hardware to generate a response versus the time it takes to compute the response via the model. This timing difference can be used as a source of secure authentication. The protocol is shown in Figure 1-1 and discussed in Section 1.1.4 [107, 130].

Such a PPUF system has widespread application. The PPUF hardware contains ***no secrets***. The PPUF hardware is a secure authentication source, but the associated model describes everything there is to know about the PPUF hardware (including all manufacturing variation). Counterintuitively, the PPUF hardware is still capable of securely authenticating itself to any server. The server also contains no secret information. Simply put, there is *no secret information anywhere in the protocol.* The authentication capability derives solely from the computational difference between the PPUF hardware and the software model, and the unclonability of the hardware (two devices with the same behavior cannot be manufactured).

In the case of a PPUF, there are no bits to steal. The security is instead based on the difficulty of reproducing an exact copy of the PPUF hardware. This represents a fundamental shift in security paradigms. Using this mechanism, a secure embedded system can be deployed in a highly untrusted environment with a strong threat model (an adversary already has access to both the PPUF design and PPUF model) and still act as a trusted authentication source.

As a result, these systems can be used in military and commercial mobile systems that must authenticate themselves to a central server or network. With the currently used approach, such devices can be spoofed if their secret information is extracted. Since the devices themselves are inherently mobile, this has always been a threat. However, were these devices to use PPUF technology, even if the devices themselves

were captured, there would be no secret information to extract. The authentication capability would remain bound to the device.

The requirements for a PPUF system can be enumerated as follows:

1. Unclonability: Impractical to manufacture two copies of the same PPUF hardware that have the same challenge/response relationship.
2. Large number of challenge response pairs: Enough challenge response pairs such that complete enumeration via model computation is impossible. (the number is ideally exponential)
3. Digital stimulation and readout.
4. Model with properties:
   (a) Correctly approximates the challenge/response relationship of the PPUF.
   (b) Requires *measurably* more time to compute than the PPUF hardware *including* stimulation/readout time.

The primary focus of this work will be on properties 2-4 of the PPUF system. Property 1 has already been studied extensively in the context of several physical systems [74, 121, 64]. Further, the notion of unclonability has been formalized in a sufficiently general manner in [10] for use in PPUF systems, although [10] does not mention PPUFs.

Therefore, this work will be focused on the primary challenge of ensuring that a PPUF is measurably faster than any software simulation (4b), while maintaining the properties that allow a PPUF to be usable in a physical system (2, 3, 4a).

5.1 Previous Work

5.2 Proposed PPUF Architecture: SIMPL

Rührmair, Devadas, and Koushanfar survey existing implementations and applications of PUFs [166]. Emerging PUF concepts such as PPUFs are discussed and surveyed. The first of these proposed architectures are SIMPL systems proposed by Rührmair et. al. in 2009 [135, 42, 37, 130, 134, 36, 136, 133]. They have discussed us-

ing a modified SRAM architecture as well as "Cellular Non-Linear Networks" (CNN).

5.2.1 Cellular Non-Linear Networks

Cellular Nonlinear networks are effectively an n (in the case of SIMPL, $n = 2$) dimensional array of analog cells. Each cell behaves according to some linear ODE as a function of its neighboring cells as well as some per-cell input parameter. Intuitively, consider a finite-difference time-domain solver where each node in the mesh may have a different first-order ODE.

While it may be the case that such systems have complex behavior that is difficult to simulate directly, the substrate chosen for SIMPL systems is CMOS. Therefore, several fundamental problems immediately present themselves.

First, the minimum internal timescale of the system's internal differential equations is limited by the speed of the analog components. In order to have a clean ODE simulation for each cell, Rührmair et. al. propose cells based on an op-amp configuration [135]. In doing so, one abstracts away from the fundamental physical equations governing analog electronics. This has two repercussions. It decreases the overall speed of the PPUF device computation and more importantly, since op-amp circuits can be tuned to behave according to any differential equation, it means that once the model is released, an analog simulator can be easily built. This fact is recognized by the authors.

According to this paper, CNN's can theoretically achieve a speedup of roughly $1000\times$ over traditional CMOS computers for specialized applications. However, with the above observation, an analog neural network could be copied with slightly slower internal dynamics, so the speedup of such a system would be significantly less than $1000\times$ against an attacker with significant resources to devote to breaking the PPUF system.

5.2.2 Modified SRAM

This implementation consists of an SRAM that introduces some errors in read/write operations. A series of pseudorandom data is written and subsequently read [135]. The concept is that a model for this SRAM can exist, but the read/write sequences would require less time than running a software model.

This assumption can only work if one is simulating the modified SRAM in software. It does not appear to be the case that there would be any speedup of the above system over an attack implemented in an FPGA or ASIC.

5.3 Proposed PPUF Architecture: FPGA Time-bounded Unclonable Authentication

Majzoobi, Nably, and Koushanfar independently presented a time-bounded authentication scheme similar to SIMPL in 2010 [103]. This system uses FPGA cells as the delay elements, and a challenge is the configuration of delay elements to produce an overall circuit. This circuit is then sampled for setup/hold time violations by a sample/capture circuit described in the paper. The security argument is simply that it takes significantly more time to simulate the FPGA in software than it does to actually run the FPGA.

The problem with this is that the delay element values are public. It is possible to scale the FPGA delays by approximately $10-100\times$ using delay lines such that you no longer have manufacturing variability, and then scale the clock frequency accordingly. Such a system can be implemented in an FPGA or ASIC platform.

5.4 Summary of Analysis of Existing PUF Solutions and PPUF Architectures

The above proposals each write out the PPUF protocol and propose hardware that is claimed to exhibit the required 'speedup' over a model such that the PPUF hardware

would be distinguishable from the model in the PPUF authentication protocol. However, as shown above, the above claims of 'speedup' for each of the proposed hardware devices is too narrow. They each claim to be faster than software implemented on a general-purpose CPU. While this may be true, this does not translate to a sufficiently strong threat model, as an adversary would not restrict their efforts only to software models. Once an adversarial hardware component is introduced, each of the above three architectures no longer demonstrate sufficient speedup to be secure.

This recognition demonstrates the fact that the demonstration of the PPUF hardware 'speedup' over the PPUF model is a critical unsolved step in the development of PPUF systems. The above proposals also provide some initial direction to potential avenues of approach. First, it is apparent that the PPUF hardware should not be created entirely from CMOS technology. One of the primary problems above is that the PPUF hardware is CMOS, and an adversarial model is CMOS. It is impossible for CMOS to exhibit a computational speedup over itself.

In addition, by analyzing the potential adversarial methods to break the security of the above proposals, it becomes clear that an adversary with CMOS hardware represents a strong security model. This is due to the fact that CMOS technology represents the culmination of trillions of dollars of investment and therefore presents by far the fastest general-purpose computational platform. If any non-CMOS PPUF hardware can demonstrate a speedup over CMOS hardware for a particular problem, then this will be due to the fact that the particular problem in question maps very well to the internal physics of the system. Any 'model' of this PPUF hardware must solve the same problem. Therefore, if a non-CMOS adversarial hardware model existed that was *also* better than the CMOS hardware model, then it must also have internal physics that maps well onto the particular problem. It is very unlikely for this to be the case unless the non-CMOS adversarial model obeyed the same physics as the PPUF hardware. In that case, the 'model' wouldn't be considered a 'model'. Rather it would be a "clone" – which is considered in a separate PPUF requirement.

In conclusion, existing PPUF proposals help define a good starting direction for the study of potential PPUF systems. They demonstrate that the PPUF hardware

should not be CMOS electronics, and that the adversary to be considered in the PPUF security model should be an adversary that can fabricate any CMOS analog/digital hardware or software in order to create an 'optimal' model of the PPUF hardware.

5.5 Informal PPUF Criteria

Analysis of the existing PPUF proposals using CNNs, SRAMs, and arbiter PUFs revealed that the proposals do not have a strong argument for *why* the PPUF instantiations have a measurably faster computation time than any software model (i.e., the "speedup requirement"). Those PPUF proposals that did articulate such an argument discussed did not result in sufficiently strong security models. Indeed, the speedup claimed in each of the previous PPUF proposals can be observed as the assumption that an adversary could not fabricate hardware circuitry even though the proposed PPUF required custom hardware. This discrepancy between the PPUF technology and the adversary results in an inconsistent security model. It was seen that in each case, relatively little financial and engineering investment would be required to build an optimized model capable of breaking the security of the PPUF protocol.

For this reason, this thesis proposes a stronger requirement – that PPUF hardware will compute the response measurably faster than a best-effort CMOS model (analyzed in the context of circuit complexity). Such a model would include custom-designed analog or digital hardware implemented using the latest CMOS technologies.

The notion of "best-effort" is formalized in Section 7 in terms of circuit complexity, but for now, I present an informal discussion for the purpose of building intuition.

This requirement is analyzed to produce three informal criteria that are formalized in Section 7. The criteria address (1) the "problem" to be solved by the PPUF hardware, (2) the physical origin of the speedup, and (3) mathematical requirements on the structure of the computational problem being solved by the PPUF hardware.

5.6 Informal Criterion 1: "Problem" Solved by PPUF Hardware

The PPUF hardware computes *some* function faster than the PPUF model, which will be implemented in CMOS as discussed above. The PPUF protocol is completely agnostic to which computational problem is chosen. Note that if one is to demonstrate a measurable speedup over a CMOS model, the PPUF hardware should not be implemented in CMOS. Speedup is comparative between computational modalities. One cannot claim a speedup of one computational modality over itself.

Therefore, a PPUF hardware must be comprised of some non-CMOS system. Since the PPUF protocol is agnostic to which computational problem is chosen, it is logical to choose the computational problem for which the PPUF hardware is optimally suited. To this end, recognize that the fastest "computational problem" that can be solved by a physical system is that of simulating its own time evolution according to its physical laws. In other words, the optimal choice for the PPUF computational problem is the direct simulation of the PPUF hardware.

This is to be contrasted with the approach taken in previous PPUF architectures, which use CMOS as a computational modality [135] and sometimes even use an abstract mathematical problem, not tied to the physics of the PPUF hardware [103].

Therefore, I claim: *the PPUF hardware can be an arbitrary system with some measurable state and some kind of input. The PPUF hardware will* ***only*** *evolve forward in time according to its physical laws. The PPUF model will then be a direct physical simulation of the PPUF hardware itself.*

5.7 Informal Criterion 2: Physical Origin of Speedup

A Public Model Physical Unclonable Function must be able to compute a function faster than a CMOS model. More specifically, this requirement is a requirement to

find PPUF hardware that has a computational speedup over a PPUF model that simulates the physics of the PPUF hardware. The search for new computational modalities that may be faster than classical computers (CMOS) is not new. Indeed, in 1982 Richard Feynman made the argument that a quantum system would be exponentially faster than a classical computer simulating it [57], simulating research in the field of quantum computing.

5.7.1 Quantum Systems

A "Universal Quantum Computer" capable of efficiently solving all problems in the BQP complexity class is believed to be difficult to build with modern technology, and as such would meet the speedup requirement for use as a PPUF [113]. However, a universal quantum computer has been found to be difficult to construct, and there is still no consensus that an experimentally implemented quantum computer has yet demonstrated a quantum speedup [113, 50].

However, a PPUF does not need universal computation, and one approach that has been taken to address the difficulty of implementing quantum computers is to limit the quantum computing model. Such an intermediate system includes the architecture of Quantum Linear Optics [5]. Quantum Linear Optics has received significant theoretical investigation [5] as well as initial experimental implementations [29, 161], and therefore represents a compelling possible approach to constructing a PPUF.

Quantum Linear Optics

Quantum Linear Optics was originally investigated by Aaronson and Arkhipov in 2010, who proposed the quantum non-interacting boson model of computation [5, 56].

They recognized that with n identical photons and m possible modes where ($n \leq m \leq \text{poly}(n)$), a computational basis state can be described as $|S\rangle = |s_1, \ldots s_m\rangle$ where s_i is the number of photons in a given mode. During a computation, photons are never created or destroyed.

This system is most conveniently described using creation/annihilation operators

(or formal variables and polynomials). Consider $X = (x_1, x_2 \ldots x_m)$ being a set of creation operators for specific modes. One can describe a state as $X^S = (x_1^{s_1} \ldots x_m^{s_m})$, where $S = (s_1, s_2 \ldots s_m)$ with $s \in \mathbb{N}$ and $\sum_i s_i = n$ for n photons.

Let U be a $m \times m$ unitary operator (mapping modes to modes). If the starting state is x^T (T defined similarly to S above), and a unitary operator U is performed on the initial state x^T, then the probability of observing state x^S is defined in terms of a submatrix of U. This matrix, defined as U_{ST}, is constructed as follows. First, a $m \times n$ matrix is constructed from U by taking t_j copies of the j'th column of U. Then, construct U_{ST} by taking s_i copies of the i'th row. Note that this submatrix corresponds to selecting the only input/output modes that matter (i.e., the ones where there are photons). The probability of observing x^S is then:

$$|\left\langle x^S | U x^T \right\rangle|^2 = |\mathrm{Per}(U_{ST})|^2$$

In other words, consider an experiment where one starts with an input state x^T, defining the locations of the n photons in the m modes. Then, apply the unitary transform U and measure the output x^S. The probability of measuring a given output x^S depends on the Permanent of a submatrix of U.

A well-known result by Valiant is that the problem of exactly computing the permanent of a matrix is #P-hard [167]. Using this result, Aaronson and Arkhipov prove that approximating $|\mathrm{Per}(A)|^2$ for $A \in \mathbb{R}^{(n \times n)}$ is not possible in polynomial time, since an oracle capable of this would result in being able to exactly compute $\mathrm{Per}(A')$, contradicting the proof that this problem is #P-hard.

Using this result, one can argue that the BosonSampling problem is hard generally as follows. Assume that there is a fast BosonSampling algorithm. Then, embed an arbitrary (real) $n \times n$ problem matrix in a larger $m \times m$ orthonormal matrix, and run the BosonSampling oracle. This provides samples from a distribution whose expectation value is the magnitude-square of the matrix permanent. The result of [154], the estimation of this expectation value to within a multiplicative factor can be computed in $\mathsf{BPP}^{\mathsf{NP}}$. However, Valiant's result in [167] is extended by Aaronson

and Arkhipov to show that such an approximation is in #P. Therfore $\mathsf{P}^{\#\mathsf{P}} \subseteq \mathsf{BPP}^{\mathsf{NP}}$. Then, the theorem of Toda implies that the polynomial time hierachy collapses to the third level [163]. This is believed not to be the case.

Application of Linear Quantum Optics to PPUFs

This problem has well defined inputs and outputs, and therefore can readily be applied to a PPUF system. A PPUF interface has an input challenge and and output response. Define the following protocol:

1. Construct a BosonSampling computer with internal unitary matrix U. Measure U and publicly post it as the 'model'. Provision BosonSampling computer to a prover.
2. A verifier picks a random x^T, and uses the "model" U to compute a distribution for x^S. This computation is time-consuming, but feasible.
3. The verifier sends T to the prover and requests the distribution of S.
4. The prover uses its BosonSampling computer to compute a distribution of x^S. Returns S.
5. The verifier determines if S is valid (by using the method of [154] to estimate the actual permanent from this distribution) and that the prover responded quickly enough to show that a model could not have been used.

Note that the calculation of the output distribution of S for a random T should be asymptotically slower for a model than for the BosonSampling computer. A linear quantum-optical system could sample from this distribution in polynomial time, while a classical computer could not. The distribution of samples can then be verified by the challenger.

However, the verification procedure does not run in expected polynomial time (from [154], it is in the complexity class $\mathsf{BPP}^{\mathsf{NP}}$). Further, the number of samples required by the verifier to accurately estimate the matrix permanent (and therefore verify that the BosonSampling computer is operating correctly) may also be exponential [154]. Therefore, even though each sample of the BosonSampling computer may have some asymptotic speedup, the overall protocol must run in exponential time.

As a result, there is no asymptotic speedup of the BosonSampling computer over a classical adversary in this protocol. Any speedup must be a constant factor.

Further, there is a significant theoretical challenge with this approach outside of the unproven conjectures put forth by Aaronson and Arkhipov. This is due to the fact that the input state is random, and that in the PPUF usage model, the challenge/response pairs are transmitted over an open channel. Therefore, multiple challenge/response pairs could be observed by the adversary prior to an attack. An adversary, given the distribution of S for a given T, might be able to compute S' for T' if T and T' are "close enough."

To understand this potential vulnerability, recognize that a randomized input corresponds to a randomized allocation of photons in the input modes. This in turn, corresponds to a different U_{ST} described above. The output distribution depends on $\mathrm{Per}(U_{ST})$. Now, consider two input challenges T, and T', where T' differs from T by only one photon placement. U_{ST} is therefore highly related to $U_{ST'}$. In fact only one row and one column would differ. Therefore, it is not clear that if an adversary is given $\mathrm{Per}(U_{ST})$, that computing $\mathrm{Per}(U_{ST'})$ is still difficult.

The difficulty of this "Differential Permanent" computation has not been investigated, and is currently an open problem.

In addition to the above theoretical challenge, it turns out that the experimental challenges associated with constructing such a PPUF using the BosonSampling problem render the approach intractable. To understand why this is the case, one needs to understand the scale needed from a BosonSampling computer in order to be useful as a PPUF.

First, Ryser's algorithm [30] solves the matrix permanent problem in $\mathcal{O}(2^{n+1}n^2)$ for a matrix with dimension n. This sets a lower-bound on the number of photons required for an implementation of BosonSampling that would be faster than a CMOS implementation of Ryser's algorithm. Assigning $n = 20$ (20 photons) corresponds to 800 million floating point operations. This would correspond to a CMOS computation time on the order of seconds. This order of computation time would be required to distinguish a linear quantum optical device from a model in the presence of network

latency on the order of 100ms. Therefore, one would likely need at least 20 photons for the systems to be distinguishable through timing.

Next, one can only measure single photons, so it is desirable to have high probability of only outputting one photon per mode. This translates to a requirement of $\Omega(n^2)$ modes ($n = 20 \rightarrow m = 400$). If one could accept multiple photons per mode, smaller m values might be acceptable. However, it remains unproven whether the BosonSampling problem remains hard with $m = \mathcal{O}(n)$. Finally, note that since the input is random, all m input and output modes are equally likely.

In summary, at least 20 photons across 400 modes would be needed with a unitary matrix U having sufficient complexity to make all possible input challenges difficult to compute.

Unfortunately, generating identical photons is still a major technical hurdle. Currently, cutting edge methods are able to generate 3 identical photons [29, 161]. However, the approach in [161] is also probabilistic, and the probability of generating three identical photons is low. It is also true that the probability of generating $n > 3$ photons decreases with n. It is not clear whether the probability of generating n photons will be exponentially suppressed. As shown above, a PPUF implementation would require at least 20 photons before any speedup over classical computers would be observable when accounting for network latency. It does not appear that such a goal is feasible in the foreseeable future.

In addition, it is not clear that the matrix U can be randomly generated in the means required by a PPUF (similar to manufacturing variability). There is currently no known way to manufacture a device with a corresponding U matrix with the size and complexity required for PPUFs. Current experiments have created simple systems, and there is no clear path to constructing complex systems in this manner that would meet PPUF requirements.

Conclusion

Basic calculations show that for a physical implementation of quantum linear optics to work as a PPUF and have measurable speedup over a classical system, at least 20

identical photons would have to be generated. This is not feasible in the foreseeable future, so one concludes that quantum linear optics cannot serve as a PPUF platform in the foreseeable future.

Remembering that universal quantum computation is not currently experimentally feasible for a sufficient number of qubits, Quantum Linear Optics represented arguably the best, most plausible approach to PPUF implementation using quantum systems. Although BosonSampling is a relatively new area of study, the technical challenges associated with it are in many ways less demanding than those for generalized quantum computational applications. Therefore, one concludes that a practical PPUF application will not be able to leverage quantum speedups in the foreseeable future.

5.7.2 Classical Origins of Computational Speedup

Potential PPUF systems leveraging quantum technologies are likely to remain infeasible experimentally for the foreseeable future. With this in mind, one can consider classical systems that may provide the required PPUF functionality.

However, with the transition to classical implementations, one must acknowledge a fundamental paradigm shift in the expectations for potential speedups. The Classical Complexity-Theoretic Church-Turing Thesis claims that a classical system can in general be simulated efficiently by a classical computer (e.g., CMOS). Therefore, one should not expect the kind of exponential speedups observed in quantum systems. Moreover, one should recognize a causal relationship between "computational speedup" and "simulation difficulty." Therefore, to identify potential classical speedups, it is logical to study systems that are known to be difficult to simulate.

Note that the converse is not true – there are systems that are difficult to simulate for reasons other than the existence of some computational speedup. For example, chaotic systems do not represent a computational speedup, but they are difficult to simulate because noise and simulation errors are amplified.

Of the systems observed in this thesis, computational speedup can in general be derived from two different, independent aspects of the physical system: internal par-

allelism and the speed of internal system dynamics. These two concepts are discussed in general below, but will be explored in the context of real physical systems later when a physical PPUF system is proposed.

First, consider physical systems that "compute quickly" because they leverage the massive internal parallelism of physical processes. For example, recognize that classical systems can be described by partial differential equations depending on local values (Maxwell's Equations, Lagrangian/Hamiltonian Mechanics). In general, modeling approaches to this type of system make use of this local behavior by imposing a finite grid/mesh (e.g., Finite-difference time-domain – FDTD, finite element method – FEM). The difficulty of modeling the system is then derived from the fact that a single or small number of CPUs must simulate each of these mesh nodes.

In this sense, such physical systems merely trade the polynomial factor in time complexity to the same polynomial factor in space complexity. One should recognize that the time-evolution of state at each of the mesh points can be computed in parallel with each other requiring communication only between adjacent mesh nodes. Clearly, in such a fully parallel model, no asymptotic speedup can be achieved because each 'simulation step' requires constant time by the physical system, and by definition only constant time by the model.

Note of course that massive 3D parallelism of 1,000,000+ CMOS computational cores is not currently feasible, so such an "ideal" modeling approach described above is not feasible and may represent a "speedup" of sorts. However, this "speedup" is more difficult to quantify and bound. This is for two primary reasons.

First, many of the limitations in parallelism of computational structures result from communication latencies, energy dissipation, and other architectural problems that are not fundamental to the technology of the underlying computing substrate (e.g., CMOS). Better computational architecture could dramatically change these capabilities.

Second, massively parallel systems are generally reduced to high-order differential equations via FDTD and FEM methods via the definition of a discrete mesh. A current, active area of research is in the field of model order reduction, which approx-

imates the above high-order system by lower-order systems [143, 127, 27, 38, 144]. Excellent progress has been made over the past few decades. However, there does not appear to be any existing theoretical methods for bounding in general how well a given system may be approximated by lower order models or in identifying "optimal" lower order models for approximation.

Therefore, using massive internal parallelism as a physical origin of speedup will be problematic, as there does not appear to be a strong mathematical foundation justifying the architectural limitations of parallel digital processing.

In contrast to those systems that compute quickly due to parallelism, there are many systems that "compute quickly" because they have fast internal dynamics. Note that these two options are not mutually exclusive. However, consider a system that is entirely sequential in nature (e.g., a system described by a single ODE). Simulating such a system becomes less about optimizing a parallel architecture, and more about comparing the simulation step computation time to the time evolution of the physical system.[1] As above, an asymptotic speedup isn't possible. However, if the time evolution of the system is fast enough with respect to the fundamental components of CMOS, then this *constant factor speedup* can be bounded.

This approach is more tractable, because the point of comparison is the individual simulation step. For the model, this simulation step would be computed by a combinatorial circuit or analog circuit corresponding to the fundamental mathematical operations associated with the simulation step. These mathematical operations are fundamental units of computing architecture (i.e., an adder or multiplier). These units have been heavily studied and optimized in analog and digital domains. As such, their performance is bounded entirely by CMOS fabrication capabilities and are therefore much less likely to dramatically change.

In conclusion, I claim: *The PPUF hardware internal dynamics must be faster than the dynamics of the CMOS model.*

There are a few caveats to this particular criterion, which are discussed and formal-

[1]Still, one must show that you can't parallelize the problem in time (i.e., parallelize the computation of intervals $[t_0, t_1)$, $[t_1, t_2), \ldots$) across multiple processors and combine with some post-processing.

ized in the following chapter. First, some sequential operations may be parallelized in time. E.g., consider an initial value problem for a linear differential equation evolving from time t_s to t_f. Although the system evolves only in time, the linear nature affords the system an impulse response at each point in time. One can then parallelize this problem by computing n impulse responses of length $(t_f - t_s)/n$ across n processors and combine the impulse responses in a $\log n$-depth tree.

Ultimately, it will be seen that the enforcement of sequential computation must be based on a conjecture. This is due to the fact that the above problem is deeply related to the comparison between the class of log-depth circuits NC, and polynomial time algorithms P. The question of whether NC = P is currently a major open problem for the complexity community.

Finally, note that the comparative notion for hardware internal dynamics is also not formalized above. This is also addressed mathematically in the following chapter.

5.8 Informal Criterion 3: Mathematical Origin of Speedup

The final criterion pertains more abstractly to the mathematical problem of simulating the physics of the PPUF hardware. In particular, this criterion states that a general-purpose discrete processor cannot use some structure inherent to the PPUF problem to achieve an asymptotic speedup over the PPUF hardware.

The PPUF hardware will evolve in time according to some set of differential equations. The abstract mathematical problem can therefore be reduced to the simulation of these differential equations.[2]

In essence, Criterion 3 places an asymptotic lower bound on the complexity of modeling the differential equation.

To understand how to show this bound on complexity, first consider an example that fails. Consider the differential equation $y'(t) = -y(t)$ with closed-form homogeneous solution $y(t) = c\exp(-t)$. In this example, the output of the PPUF hardware

[2]The ODE must be non-chaotic and stable.

would be $y(t)$. Therefore, to compute the output at time t, the PPUF hardware would require time t. However, a model can compute this value in asymptotically less time. This is due to the fact that $\exp(t)$ can be numerically approximated at time t in $O(\log(t))$ by using repeated squaring for the integer component of t and using a precomputed interpolation of $\exp(t)$ for $0 < t < 1$ for the fractional component of t. This example differential equation would fail Criterion 3 because the function $\exp(t)$ can be evaluated numerically sublinearly in t.

An example of a differential equation that does not fall into the trap of the example above is the Bessel equation (for some argument and precision requirement). The best-known algorithms for computing the Bessel function $J_0(t)$ are super-linear in t.[3] The fundamental difference between these two examples is that the former can be represented "in closed form." Mathematically speaking, it is a "Liouvillian Solution."

Differential Galois Theory is the field of identifying differential equations whose solutions can be represented in closed-form. In this thesis, this theory provides a succinct set of criteria for defining when a differential equation has a closed-form solution and when it does not. This theory will be used to provide strict criteria for the differential equation to ensure that it does not have closed-form solutions.

Before proceeding further, one can now impose an additional restriction on the class of differential equations that could be used to describe the PPUF hardware. First, recognize that ODEs can be linear time-invariant, linear time-varying, or non-linear. It is well known that linear time-invariant differential equations all have closed-form solutions in terms of the matrix exponential. Therefore, one can rule this class out. Second, while non-linear equations provide the most complex behavior, there is also very little structure to their classification. Therefore, Differential Galois Theory does not extend into the theory of general non-linear differential equations. One can conclude therefore, that although nonlinear equations have rich, complex behavior, current mathematical tools are not sufficiently well developed to provide adequate confidence in the difficulty of simulating non-linear differential equations.

For the above reasons, one is restricted to considering linear time-varying differ-

[3]Ignoring pre-computation, and only for small arguments and high precision.

ential equations. For a large subclass of these differential equations, the criteria put forth by Differential Galois Theory can be described by an algorithm – the Kovacic Algorithm [90, 142]. Using this algorithm, one can provide a further restriction on the class of differential equations that can be used for PPUF systems. This analysis is presented in Chapter 8. It is seen that the Kovacic algorithm by itself does not provide sufficient criteria for a differential equation whose model requires $\Theta(t_f - t_s)$ steps. There are other workarounds that need to be addressed.

Once it is shown that no closed form solution can be used to exactly or approximately represent the output of the differential equation, this thesis presents a fundamental conjecture on which PPUF security will be based (Conjecture 8.4.1). Informally, this conjecture states that *a differential equation with no exact or approximate closed form solution is best approximated using algorithms with local knowledge of the ODE (e.g., the RK4 integration method).* This thesis provides empirical evidence for this conjecture by observing the history of study of differential equations and the fact that even after this multi-century long study, optimal numerical algorithms still use analytic continuation.

Finally, one needs to bound the step size of the analytic continuation algorithm to show that it requires $\Theta(t_f - t_s)$ steps to reach the system state at time t_f. Because I assume that the expansion algorithm has only local knowledge of the ODE in the conjecture, and it is required that this expansion algorithm be computable in finite time, recognize that there is some maximal expansion order $k_{\max}$ that is always greater than the expansions used by the numerical integration algorithm. Using this bound, recognize that the expansion algorithm does not know of the behavior of the ODE at all times, and therefore by an information theoretic argument cannot compute the solution of the ODE for all time. By analyzing this property formally using differential entropy, a maximal step size can be shown.

With this in mind, Criterion 3 can be stated: *The computational problem that the PPUF hardware solves must have the property that any digital circuit has an asymptotic lower bound on time complexity of* $\Theta(t_f - t_s)$.

In conclusion, these three informal criteria represent sufficient conditions to show

that the PPUF hardware will be faster than its model by a constant factor. In Criterion 1, it is recognized that the "computational problem" being solved by the PPUF should be the time evolution of the PPUF hardware according to its physical laws. In Criterion 2, it is shown that the PPUF hardware should be classical in nature, it should have internal dynamics that are faster than CMOS, and that the PPUF problem should not be parallelizable. Finally, Criterion 3 requires that there is no asymptotic speedup for a general-purpose digital processor.

A PPUF system that meets these three criteria will meet the fundamental theoretical PPUF requirement that any CMOS model will always be slower than the PPUF hardware.

6 - Mathematical Preliminaries: Symbolic Computation

Differential Galois Theory addresses the question of which ODEs have closed form solutions, and how to compute these solutions symbolically. Differential Galois theory identifies closed-form solutions to ODEs in the same way as traditional Galois theory [12] identifies radical solutions to polynomials. Indeed, Differential Galois theory is at its core an application of traditional Galois theory to a differential field, which can be thought of as a class of ODE operators. Therefore, I first introduce several differential algebra concepts. All material covered in this section is based on the treatment in the cited sources [88, 149, 101, 79, 41, 35, 140, 46].

Note that Differential Galois Theory is only maturely developed for the class of linear time-varying differential equations (LTV ODEs). In addition the Kovacic algorithm only works for LTV ODEs with coefficients that are fractional polynomials in the indeterminate. Therefore, I restrict the class of differential equations considered for use in a PPUF to this subclass, as differential equations outside of this class in general do not appear to have a sufficient theoretical framework to provide the required guarantees that no closed-form solution exists.

6.1 Differential Galois Theory

Definition 6.1.1. *A differential field k is a field equipped with a derivation operator $\partial : k \to k$ that obeys the following property:*

1. $\partial(a+b) = \partial(a) + \partial(b)\ \forall a, b \in k$

2. $\partial(ab) = \partial(a)b + a\partial(b)\ \forall a, b \in k$

The definition of a differential ring and ideal is similar. Note that the field/ring/ideal must be closed under the derivation operator.

For the sake of this work, only fields of characteristic 0 will be considered (fields of higher characteristic do not correspond to physical systems of interest, and the Differential Galois theory is not as mature). Therefore, we consider that the base field will always be the complex numbers $\mathbb{C}$ with the derivation operator $\partial(a) = 0 \forall a \in \mathbb{C}$. Note that it will be important that the base field is algebraically closed.

6.1.1 Picard-Vessiot Extensions

Differential Galois Theory uses sets of "extensions" to this base field as a way of characterizing solutions to differential equations. To understand this, consider how to represent differential equations in the above language of differential algebra. A differential equation over the field k is represented abstractly as a "differential module."

Definition 6.1.2. *A differential module consists of an element M_n is an n dimensional k-vector space equipped with $\partial : M \to M$ which obeys the property: $\partial(fm) = f'm + f\partial(m)\ \forall f \in k, \forall m \in M$.*

More concretely, if M_n has dimension n, then it has a basis of e_i, with $i \in [n]$. One can then represent any M_n by how ∂ acts on its basis:

$\partial e_i = -\sum_j a_{j,i} e_j$

Note that the minus sign is of historical significance and is of no importance. The matrix $A = (a_{i,j})$ with each $a_{i,j} \in k\ \forall i, j$ completely defines the behavior of M_n. Next, one defines $y = (y_1, ..., y_n)^T \in k^n$ where ∂ extends to matrices/vectors component-wise. Finally, the differential equation can be represented as:

$y' = Ay$

Of course, one could always pick a different basis for the vector space defined by M_n. Therefore, any $B = AC, C \in \mathrm{GL_n}(k)$ is also in $M_n(k)$. Going forward, only a

single basis needs to be considered, so a differential module M_n can be thought of as equivalent to a matrix A that defines the differential equation as shown above.

Next, one can consider the solutions to the differential equation as a vector space as well:

Definition 6.1.3. *Let R be a differential ring containing the differential field k. Let $A \in M_n(k)$. An invertible matrix $F \in \mathrm{GL_n}(R)$ is a fundamental matrix for equation $y' = Ay$ if $F' = AF$.*

Once again, more concretely, F can be thought of as a set of possible solutions to the differential equation. Note that the elements of F are in R, and not necessarily in the underlying field k. This is because the solution to the differential equation may not exist in the underlying field k. For example, if $k = \mathbb{C}$, and the differential equation is $y' = -y$. The solution $e^{-t} \notin \mathbb{C}$. (Note that t is the 'independent variable' here. In the formulation of differential algebra, one can abstractly discuss differentials without ever describing the independent variable. Rather one can think of e^{-t} as some $a \in R$ such that $\partial a = -a$ without ever worrying about t).

Regardless, the solution is not in $k = \mathbb{C}$, so one must extend the field k to include the solution. More generally, one can say that for many differential equations, the associated fundamental matrix $F \notin \mathrm{GL_n}(k)$.

Definition 6.1.4. *A Picard-Vessiot ring over k for the equation $y' = Ay$ is a differential ring R over k satisfying:*

1. *R is a simple differential ring (the only proper differential Ideal is $\{0\}$).*
2. *There exists a fundamental matrix F for $y' = Ay$ with coefficients in R.*
3. *R is generated as a ring by k, the entries of a fundamental matrix F, and the inverse determinant of F.*

Concretely, this means that the Picard-Vessiot ring contains all of the solutions of the differential equation and is minimal (i.e., it doesn't contain any redundant solutions. If redundant solutions were contained, then dividing them would generate

an extra constant value, which would then form a differential ideal when combined with other constants).

Example 6.1.1 Let the base field be $F = \mathbb{C}(x)$ equipped with differential D such that $D(c) = 0\ \forall c \in \mathbb{C}$ and $D(x) = 1$. Compute the Picard-Vessiot extension for $y'' = y$.

Solution. The above differential equation is a simpler example because the solution can be represented in closed form as exp. A complete set of solutions of the above differential equation is $[\exp(x), \exp(-x)]$.

Now, consider the field extension $E \supset F$, where $E = F[\exp(x), \exp(-x)]$. Because $\exp(-x)$ is the inverse of $\exp(x)$, the above extension is already a field (i.e., it is already the field of fractions of $\exp(x)$). Clearly it contains all of the solutions of $y'' = y$. Finally, recognize that it does not add any new constants. A proof can be found in both [149, 101]. However, the intuition is that the extensions $\exp(x)$, $\exp(-x)$ cannot be used to derive any new constants because the only way that $\exp(x)$ can be combined with *any* other elements to obtain a constant is through its inverse, $\exp(-x)$.

Therefore, the Picard-Vessiot extension of $y'' = y$ is $E = F(\exp(x))$. ◀

Definition 6.1.5. *A Picard-Vessiot field for a differential equation is the field of fractions of the Picard-Vessiot Ring for the same equation.*

Now that the Picard-Vessiot field has been defined, this field can be understood as a field extension of the base field. For example, with the base differential field $F = \mathbb{C}$ (with $\partial(c) = 0\ \forall c \in \mathbb{C}$), and a Picard-Vessiot ring $\mathbb{C}[x]$ where $\partial(x) = a$ with $a \in \mathbb{C}$. Let E be the field of fractions of F, and therefore be the associated Picard-Vessiot field. One can now think of $E \supseteq F$ as a field extension and apply the notions of Galois theory.

6.1.2 Differential Automorphisms

Remember that the above field is defined such that it is the minimal field extension such that all of the solutions of a given differential equation $y' = Ay$ exist within the

new field. Therefore, one can logically think about the degree of the field extension as the dimension of the vector space containing the solutions. In the case of a n'th order differential equation, this corresponds to an n degree vector space. One can intuitively see this based on the fact that any general n'th order differential equation requires n initial conditions to provide a unique solution. Therefore, there are n parameters that can be modified.

Definition 6.1.6. *Let $E \supseteq F$ be a differential field extension. The set* $\mathrm{Gal}(E/F)$ *of automorphisms $\sigma : E \to E$ such that $\sigma(\partial(a)) = \partial(\sigma(a))\ \forall a \in E$ and $\sigma(a) = a$ $\forall a \in F$ is called the Differential Galois Group of E over F. The group operator is the composition of the automorphisms. As above, the group is denoted* $\mathrm{Gal}(E/F)$.

A key feature of the Differential Galois Groups is that any given element $\sigma \in \mathrm{Gal}(E/F)$ can be represented as an element of $\mathrm{GL}_n(C)$, where n is the dimension of the extension (equivalent to the order of the differential equation) and C is the set of constants in F. Note that by the definition of a Picard Vessiot field, C must also be the same set of constants in E as well. For the purposes of this discussion, $C = \mathbb{C}$.

Before this discussion, recognize that a complete set of solutions to a differential equation is represented by a fundamental matrix F such that $F' = AF$. Clearly, right multiplication by any $M \in \mathrm{GL}_n C$ does not affect the validity of the solution: $(FM)' = AFM$. It can be shown that this represents the complete set of fundamental matrices (i.e., the solution space is linearly independent over the constants). Therefore, if one considers $\sigma(F)$, it can be seen that $\sigma(F)' = \sigma(F') = \sigma(AF) = A\sigma(F)$. Therefore, $\sigma(F)$ is still a fundamental matrix. Therefore, the σ operator can be represented as a right-multiply by a certain constant matrix. It is clear, therefore, that $G = \mathrm{Gal}(E/F)$ can be represented as a subset of $\mathrm{GL}_n(C)$.

Note that although the above matrix representation holds, the restrictions on the possible automorphisms does not come from finding the subset of M such that $(FM)' = AM$. This is true for all $M \in \mathrm{GL}_n(C)$. Rather, the restriction is that for each $e \in E$, $\sigma(e)' = \sigma(e')$. In addition, the automorphism must leave the base field fixed elementwise and preserve the 'solution' space V where $E = F\langle V \rangle$. The purpose

of recognizing that $G \subseteq \mathrm{GL}_n(C)$ is to restrict the possible set of groups for G to take and to direct and understand how G is composed. In doing so, it will be shown that this group structure will determine whether or not the solutions will be representable in closed form.

The first step is to recognize that G must in general be isomorphic to an algebraic subgroup of $\mathrm{GL}_n(C)$. This is proved in [149] and a proof sketch is given below. An algebraic subgroup of an algebraic group (like the differential Galois group) is called a Zariski closed subgroup.

Definition 6.1.7. *A subgroup K of $\mathrm{GL}_n(C)$ is an algebraic group if there exist a finite number of polynomials $P_1, P_2, \ldots P_m$ where each $P_i \in C[X_1 \ldots X_{n^2}]$ such that a matrix $(x_{i,j})$ is an element of K iff $P_i(x_{1,1}, x_{1,2}, \ldots x_{n,n}) = 0$ for $1 \leq i \leq m$.*

This can be seen intuitively as follows. Recall that the Picard-Vessiot field $E \supseteq k$ for a differential equation described by $y' = Ay$ is given by the fraction field of $R = k[x_{i,j}]\left[\frac{1}{\det}\right]/q$, where $F = (x_{i,j})$ and $\det = \det(F)$. Also, q is a maximal differential ideal of $k[x_{i,j}]\left[\frac{1}{\det}\right]$.

Recall from above that differential ideals of the above ring represent 'redundancy' in the solution. Magid proves that, if the base field k is algebraically closed (which it is in this case, $C = \mathbb{C}$), then the existence of a differential ideal implies the addition of new constants [101]. By taking the quotient group, these 'new' constants are effectively assigned to existing constants. In doing so, this means that any k-automorphism must fix all elements of the ideal, since the quotient group 'assigns' them to values in k, and k is fixed elementwise. Specifically stated, $\forall \sigma \in G, \sigma(q) \subseteq q$.

To see what requirements this defines, let $q_1, \ldots q_r$ denote the generators of the differential ideal q, and let $\{e_i\}$ be a C-basis of R. This means that the initial requirement $\sigma(q_j) \mod q = 0$ can be expressed as a finite sum $\sum_i C(M,i,j)e_i = 0$. $M \in \mathrm{GL}_n(C)$ is the matrix corresponding to the automorphism σ. $C(M,i,j) \in C$ are constant coefficients depending on M, indexed by i, j. Thus, the finite set of equations is given by $\{C(M,i,j) = 0\}_{i,j}$. Therefore, σ is an algebraic subgroup of $\mathrm{GL}_n(C)$.

This general fact is about all that can be said about the n'th order general case. More specific work can be performed if certain assumptions are made or if a certain differential equation is given.

Example 6.1.2 Above, the Picard-Vessiot extension for $y'' = y$ is calculated as $E = F(\exp(x))$ where $F = \mathbb{C}(x)$ with the normal derivation operator D. Now, calculate the group $\mathrm{Gal}(E/F)$.

Solution. Remember that the condition for the differential Galois group above is that it must commute with the derivation operator. That is, for $\sigma \in \mathrm{Gal}(E/F)$, $\sigma(x)' = \sigma(x') \; \forall x \in E$. The set $[\exp(x), \exp(-x)]$ represents a solution basis (each element is linearly independent). Therefore, one can consider the operation of $\sigma \in \mathrm{Gal}(E/F)$ on $exp(x)$ and $exp(-x)$ and recognize that since σ is a field automorphism, that the remaining add/multiply operations follow.

Specifically, remember that for a second order differential equation, $\mathrm{Gal}(E/F) \subset \mathrm{GL}_2((C))$. Therefore, the following can be considered to be true without loss of generality:

$\sigma(\exp(x)) = a\exp(x) + b\exp(-x)$

$\sigma(\exp(-x)) = c\exp(x) + d\exp(-x)$

where $a, b, c, d \in (C)$. Now, consider the differential commutation property for the first equation above:

$\sigma(\exp(x)') = \sigma(\exp(x)) = a\exp(x) + b\exp(-x)$

This must equal:

$\sigma(\exp(x))' = a\exp(x) - b\exp(-x)$

Therefore:

$a\exp(x) + b\exp(-x) = a\exp(x) - b\exp(-x)$

This implies that $b = 0$ and a is unconstrained. Doing the same for the second equation results in $c = 0$ similarly. However, recognize that σ must maintain the algebraic properties of the field. Therefore, one can establish a relationship between a and d by recognizing that $\exp(-x) = \exp(x)^{-1}$. Therefore, $a = d^{-1}$. Therefore, the Galois group can be represented by:

$$\sigma \in \left\{ \begin{pmatrix} a & 0 \\ 0 & a^{-1} \end{pmatrix} \text{ s.t.} a \in \mathbb{C} \right\}$$

◀

Example 6.1.3 Calculate the Picard-Vessiot extension and associated differential Galois group for the differential equation $y'' = -\frac{2}{9x^2}y$.

Solution. The above differential equation has a solution space $\left[x^{1/3}, x^{2/3}\right]$. Therefore, one can immediately recognize the Picard-Vessiot field extension as $E = F(x^{1/3})$ using the same intuition as in the previous example. Once again, this is not a formal proof.

To compute the automorphism group the same approach is used. Assign $\sigma(x^{1/3}) = ax^{1/3} + bx^{2/3}$ and $\sigma(x^{2/3}) = cx^{1/3} + dx^{2/3}$. Recognize that $\sigma(x)' = \sigma(x')$. Applying this to the two above linearly independent solutions and simplifying results in the following two equations:

$\left(d + \frac{c}{2}x^{-1/3}\right) \cdot \left(a + bx^{1/3}\right) = 1$.

$\left(2b + ax^{-1/3}\right) \cdot \left(c + dx^{1/3}\right) = 1$

Now, since the right hand side does not depend on x, this translates to four non-redundant equations:

$da + \frac{bc}{2} = 1$

$db = 0$

$ca = 0$

$2bc + ad = 1$

This corresponds to $b = c = 0$ and $a = \frac{1}{d}$. Therefore, one can conclude that the differential Galois group for the differential equation $y'' = -\frac{2}{9x^2}y$ is:

$$\sigma \in \left\{ \begin{pmatrix} a & 0 \\ 0 & a^{-1} \end{pmatrix} \text{ s.t.} a \in \mathbb{C} \right\}$$

This is the same group as the previous example.

◀

6.1.3 The Differential Galois Correspondence

Now that the general structure of differential Galois groups is better understood, the differential Galois correspondence can be stated. This correspondence is analogous to the traditional Galois correspondence in the case of polynomials.

This process begins by recognizing that $g \in \mathrm{Gal}(E/k)$ leaves any ideal q invariant. Therefore, define $Z = \max(R)$ where R is defined as above (Note: $\max(X)$ is the set of all maximal ideals of X). It is also shown in [149] that Z is a reduced, irreducible subspace of $\mathrm{GL}_n(k) := \max(k\left[X_{i,j}, \frac{1}{\det}\right])$. Therefore, $g \subset \mathrm{GL}_n(C)$ such that $Zg = Z$. Note that multiplication on $\mathrm{GL}_n(k)$ induces the morphism $m : Z \times_C G \to Z$ given by $(z, g) \mapsto zg$. This morphism is also a group action when considering multiple $g_i \in G$.

Using this fact, it is also proved that the morphism $Z \times_C G \to Z \times_k Z$ given by $(z, g) \mapsto (zg, z)$ is an isomorphism of affine varieties over k. This is the definition of a torsor, so one has that Z is a G-torsor over k.

One recognizes that if Z is a G-torsor over k, extensions to k affect the properties of Z. Specifically, if Z contains a k-rational point p (denoted $Z(k) \neq \emptyset$), then the relationship between G_k and Z collapses to $Z = pG_k$, and Z is called a trivial torsor. Through some work, one can prove that a finite extension of k will result in the torsor becoming trivial, and the transcendence degree of this extension is the dimension of G. In addition, one finds that if H is a subgroup of G with algebraic closure $\overline{H}$, then $E^H = k$ if and only if $\overline{H} = G$. Note that E^H is $\{e \in E : \sigma(e) = e \, \forall \sigma \in H\}$.

Thus, one begins to build a correspondence between the differential Galois groups and the associated differential field extensions in the same way as traditional Galois theory. The following Galois correspondence can be established (proof in [149]):

Theorem 6.1.8. *Let $y' = Ay$ be a differential equation over k with Picard-Vessiot field E and let $G := \mathrm{Gal}(E/k)$. Define two sets:*

$\mathcal{S} :=$ the closed subgroups of G

$\mathcal{L} :=$ the differential subfields M of E containing k.

Define $\alpha : \mathcal{S} \to \mathcal{L}$ by $\alpha(H) = E^H$, the subfield of E consisting of H-invariant elements.

Define $\beta : \mathcal{L} \to \mathcal{S}$ *by* $\beta(M) = \mathrm{Gal}(E/M)$, *the subgroup of* G *consisting the* M*-linear differential automorphisms.*

The following properties hold:

1. *maps* α *and* β *are inverses of each other.*

2. *The subgroup* $H \in \mathcal{S}$ *is a normal subgroup of* G *if and only if* $M = E^H$ *is invariant under* G *as a set. If* $H \in \mathcal{S}$ *is normal, then the canonical map* $G \to \mathrm{Gal}(M/k)$ *is surjective and has kernel* H. *Moreover,* M *is a Picard-Vessiot field for some linear differential equation over* k.

3. *let* G^o *denote the identity component of* G. *Then* $E^{G^o} \supset k$ *is a finite Galois extension with Galois group* G/G^o *and is the algebraic closure of* k *in* E.

6.1.4 Liouvillian Extensions

The above differential Galois correspondence allows the study of the forms of solutions of differential equations in terms of the group structure of the associated differential automorphisms of the field extension. In particular, it can formalize the notion of 'solving a linear differential equation in finite terms'. This type of solution would be a algebraic combination of exponentials, integrals, and radicals.

In particular, these 'closed form' solutions are represented as elements of 'Liouvillian Extensions' to the original differential field. Consider the following definition:

Definition 6.1.9. *The differential field* k *with an algebraically closed field of constants* C *is extended to* $K \subset k$. K *is called a Liouvillian Extension of* k *if there exists a tower of fields* $k = K_0 \subset K_1 \subset \cdots \subset K_n = K$ *such that* $K_i = K_{i-1}(t_i)$ *for* $i = 1, \ldots, n$ *where one of the following is true:*

1. $t_i' \in K_{i-1}$. *This means that* t_i *is an integral of an element of* K_{i-1}.

2. $t_i'/t_i \in K_{i-1}$. *(Note* $t_i \neq 0$*). In this case,* t_i *is the exponent of an integral of an element of* K_{i-1}.

3. *t_i is algebraic over K_{i-1}. In this case, t_i is a radical of one of the elements of K_{i-1}.*

Using this definition, one can state the primary result of Differential Galois theory as it pertains to closed-form solutions to differential equations:

Theorem 6.1.10. *Let E be a Picard-Vessiot extension of k with differential Galois group G. The following are equivalent:*

1. *G^o is a solvable group.*

2. *E is a Liouvillian extension of k.*

3. *E is contained in a Liouvillian extension of k.*

A full proof of the above is given in [149, 101]. At a high level, one can show (1) $\implies$ (2) by invoking the Lie-Kolchin theorem, which states that the solution space $V \subset E$ has a basis $y_1, \ldots, y_n$ over C such that $G^o \subset \mathrm{GL}(V)$ consists of upper triangular matrices with respect to the defined basis. One uses induction, showing how to incorporate y_1 as a field extension and then recognizing that one can do this iteratively through all y_i.

There are two cases: $y_1 \in k_0$ and $y_1 \notin k_0$. If $y_1 \notin k_0$, and $y_1 \notin \overline{k_0}$, then because G^o is upper triangular, there exists some constant $c(\sigma)$ for $\sigma \in G^o$. Therefore, one can write $\sigma y_1 = c(\sigma) y_1$, so therefore $\frac{y_1'}{y_1} \in k_0$. Now, consider the extension E over $k_0(y_1)$. The associated differential Galois group is a proper subgroup of G^o.

Note that if $y_1 \notin k_0$, but $y_1 \in \overline{k_0}$, then a standard algebraic extension as per traditional Galois theory suffices, and one can iterate on the remaining y_i.

If $y_1 \in k_0$, then a different approach is required. Consider the representation where the original differential equation $L(y) = 0$ has the form $y^{(n)} + \cdots + a_0 y^{(0)} = 0$. Consider $L(yy_1)$. Since $L(y_1) = 0$ and L is linear, L distributes over y and y_1 to have the form $b_n y^{(n)} + \cdots + b_1 y^{(1)} + b_0 y^{(0)} = 0$. Because $L(y_1) = 0$, one knows that $b_0 = 0$. Now, consider the differential equation $M(f) = b_n f^{(n-1)} + \cdots + b_1 f = 0$. The associated solution space in K is $\sum_i C \left(\frac{y_i}{y_1}\right)'$ where $i = 2, \ldots, n$. This space becomes

the solution space for the next step. It has all of the properties of the original solution space, except now the basis terms are $t_i = \left(\frac{y_i}{y_1}\right)'$. This induces an integration.

Next, one can show that (3) $\Longrightarrow$ (1) by considering a Liouvillian extension $M = k(t_1, \ldots, t_m)$ containing E to show that G^o is solvable. One breaks the extended field $E \subset M$ into individual extensions. Specifically, consider $K(t_1)$ as the field generated by solutions of $L(y) = 0$ over $k(t_1)$ and their derivatives. The associated Galois group $H = \text{Gal}(K(t_1)/k(t_1))$ is a closed subgroup of G. The invariant field $K^H = K(t_1)^H \cap K = k(t_1) \cap K$. Note that K is also the Picard-Vessiot field of solutions for $L(y) = 0$ over $k(t_1) \cap K$. Therefore, the associated Galois group with this subfield extension is $H = \text{Gal}(K/k(t_1) \cap K)$. One can iterate this method over y_i. As long as the extension associated with each y_i is solvable, the maximal group H^o is also solvable.

Now, consider the possibilities for each extension t_1:

1. If $k(t_1) \cap K = k$, then $H = G$.
2. If $k(t_1) \cap K \neq k$, then there are several possibilities:
   (a) If t_1 is algebraic over k, then clearly $k(t_1) \cap K$ is also algebraic over k and lies in the fixed field K^{G^o}. Therefore, $H^o = G^o$.
   (b) If t_1 is a transcendental extension where:
      i. $t_1' = a \in k^*$, then t_1 is an *adjunction of an integral*, which can be shown to be associated with the differential Galois group $\mathbf{G}_{a,C}$, the additive group over C. This group only has trivial algebraic subgroups and is abelian and is therefore solvable.
      ii. $t_1'/t_1 = a \in k^*$, then t_1 is an *adjunction of an exponentiated integral*, which can be shown to be associated with the Galois group $\mathbf{G}_{m,C}$, the multiplicative group over C. The only non-trivial closed subgroups of $\mathbf{G}_{m,C}$ are the finite groups of roots of unity. Hence, $k(t_1) \cap K$ will be of the form $k(t_1^d)$ for integer $d \geq 1$. This implies that the associated G^o is solvable.

Therefore, as shown above, Liouvillian extensions represent the solution spaces of differential equations that have closed-form solutions. Therefore, by determining the

structure of the differential Galois group associated with the differential equation, one can also determine whether or not the differential equation has closed-form solutions.

6.2 Kovacic's Algorithm

Although the above theorem suggests a clear path to demonstrating unequivocally the existence/non-existence of a closed form solution for any differential equation, there are several major difficulties. The first is that the above theorem requires linear differential equations. There is not currently a general method of extending the above theorem to nonlinear differential equations.

In addition, even for linear differential equations, the case is not clear cut. It is not known how to find the Galois group associated with a differential equation and determining its solvability property in general. This stems from the fact that in general, the lattice of algebraic subgroups is difficult to describe.

However, by restricting the scope of possible differential equations, an algorithmic approach arises that can solve large, relevant classes of differential equations. Specifically, consider second order linear differential equations where the coefficients are in $\mathbb{C}(t)$ (where t is the independent variable). E.g., $f(t)y''(t) + g(t)y'(t) + h(t)y(t) = 0$, where $f(t)$, $g(t)$, $h(t) \in \mathbb{C}(t)$.

First, one recognizes that the Wronskian of the solutions to a general second-order linear differential equation is constant and therefore must be left constant by all elements of the differential Galois group. Remember that the differential Galois group is a subgroup $G \subset \mathrm{GL}_2(C)$. The requirement of leaving the Wronskian constant translates to the requirement that $G \subseteq \mathrm{SL}_2(C)$, the group of 2×2 matrices with determinant one.

It has been shown that for the above type of differential equation, the possible Galois groups can be restricted to one of four categories [88]. Kovacic then used these four categories to show that there can only be four classes of solutions to second order differential equations of the above form [90].

Theorem 6.2.1. *The differential Galois group*[1] *G of the equation: $y''(t) = r(t)y(t)$ for $r(t) \in \mathbb{C}(t)$ is an algebraic subgroup of $\mathsf{SL}_2(\mathbb{C})$ of one of the following four forms.*

1. *G is triangularizible (can be made into upper/lower triangular matrix by basis transformation) such that the differential equation is reducible and has a solution of the form $e^{\int \omega}$ where $\omega \in \mathbb{C}(t)$.*

2. *G is imprimitive (for $\mathrm{SL}_2(\mathbb{C})$, will have 2 eigenvalues of the same magnitude) and the above equation has a solution of the form $e^{\int \omega}$ where ω is algebraic over $\mathbb{C}(t)$ of degree 2 and case 1 does not hold.*

3. *G is primitive and finite, and the above equation has an algebraic solution of the form $e^{\int \omega}$ where ω is algebraic of degree 4, 6, or 12 over $\mathbb{C}(t)$. Case 1 and 2 do not hold.*

4. *$G = \mathrm{SL}_2(\mathbb{C})$, and there are no Liouvillian solutions to the equation. Case 1, 2, and 3 do not hold.*

Using this theorem, an explicit algorithm can be given that identifies closed-form solutions to second order differential equations of this form where possible, and also identifies when this is not possible [90, 142, 72, 151].

The first step is to recognize that the above differential equation:

$$ay''(t) + by'(t) + cy(t) = 0$$

can be reduced by making the transformation:

$$y(t) = z(t)e^{-\int \frac{b}{2a} \mathrm{d}t}$$

[1]The "differential Galois group" is analogous to the traditional Galois group, and has the same relationship to the root of the associated differential operator (i.e., solution of the ODE) as a traditional Galois group has to the root of the associated polynomial. This theory is described at length in Chapter 1 of [149]. As in traditional Galois theory, if the differential Galois group is "solvable," then the corresponding solution can be represented in closed form. Theorem 6.2.1 completely enumerates all possible solvable differential Galois groups for second-order linear ODEs and their associated closed-form solutions.

such that the equation becomes:

$$y''(t) - \left(\frac{b^2 - 2ab' - 2ba' - 4ac}{4a^2}\right) y(t) = 0$$

or, more concisely, $y''(t) = ry(t)$ for $r \in \mathbb{C}(t)$. Therefore, one can consider only equations of this form without loss of generality. With this in mind, the above theorem states that there must be an equation of the form $\eta = e^{\int \omega}$. The algorithm uses the following fact several times: The equation $y''(t) = ry(t)$ has a solution of the form $\eta = e^{\int \omega}$ iff ω satisfies the Riccati equation $\omega' + \omega^2 = r$ [151].

6.2.1 The Kovacic Algorithm for Case 1

I do not include proofs of the necessary conditions. I show a proof of the algorithm only for Case 1. Cases 2 and 3 are similar, and are discussed in [90].

For each of the three possible cases resulting in Liouvillian solutions, the associated algorithm for determining a solution η is very similar. Below, the first case will be discussed without proof for the purpose of gaining intuition for the functionality of the algorithm. Full details and proofs are available in the original work by Kovacic and Saunders [90, 142].

In the first case, the most general solution can be described as $Pe^{\int \omega} = e^{\int \omega + \frac{P'}{P}}$ for $P \in \mathbb{C}(t)$. Let $\theta = \omega + \frac{P'}{P}$. The method will be to determine the partial fraction expansion of θ by using the Laurent series expansion of θ and r related by the Riccati equation above. For example, if θ has a pole of order ν at c, the expansion around c is:

$$\theta = \sum_{i=2}^{\nu} \frac{a_i}{(t-c)^i} + \frac{\alpha}{t-c} + \sum_{i=0}^{\inf} b_i t^i$$

The "Component at c" is:

$$[\theta]_c + \frac{\alpha}{t-c} = \sum_{i=2}^{\nu} \frac{a_i}{(t-c)^i} + \frac{\alpha}{t-c}$$

Let $\overline{\theta}$ be the remaining terms that are not in the component (of degree 0 and higher in t).

Remember that the pole order at c must be 1,2, or an even integer greater than 2. The algorithm calculates the "component" parameters at c in order to assemble the partial fraction expansion of θ. Depending on the order of the pole, the algorithm must use one of several methods.

If the pole of r at c is order 1, then $[\theta]_c = 0$ and the Riccati equation becomes:

$$-\frac{\alpha}{(t-c)^2} + \cdots + \frac{\alpha^2}{(t-c)^2} + \cdots = \frac{\text{Const}}{t-c} + \cdots$$

Note that all of the terms in "$\cdots$" are of order t^n for $n \geq -1$. In order for the equation to hold, the t^{-2} terms must cancel, meaning that $-\alpha + \alpha^2 = 0$, or $\alpha = 1$. Therefore, all poles in r of order 1 have a corresponding term in the partial fraction expansion of θ with coefficient 1. Note that $\alpha \neq 0$ because if this were the case, then the left-hand-side of the Riccati equation would not have a pole, and the right hand side would, which is incorrect.

If the pole of r at c is order 2, then $[\theta]_c = 0$ again, and the Riccati equation becomes:

$$-\frac{\alpha}{(t-c)^2} + \cdots + \frac{\alpha^2}{(t-c)^2} + \cdots = \frac{\text{b}^{\,2}}{t-c} + \cdots$$

For the t^{-2} terms to equal, it must be the case that $-\alpha+\alpha^2 = b$, or $\alpha = \frac{1}{2} \pm \frac{1}{2}\sqrt{1+4b}$.

If the pole of r at c is even and order $2\nu \geq 4$, then $[\theta]_c \neq 0$.

First, recognize that the Laurent series of $\sqrt{r}$ at c will involve terms $1/t^i$ for $i < \nu$ and t^j for $j \geq 0$.

Define $[\sqrt{r}]_c = \frac{a}{(t-c)^\nu} + \cdots + \frac{\text{Const}}{(t-c)^2}$. Next, define $\overline{r} = \sqrt{r} - [\sqrt{r}]_c$. Then, $r = [\sqrt{r}]^2 + 2\overline{r}[\sqrt{r}] + \overline{r}^2$. Plugging this into the Riccati equation yields:

$$([\theta] - [\sqrt{r}]_c) \cdot ([\theta] + [\sqrt{r}]_c) = -[\theta]' + \frac{\alpha}{(t-c)^2} - \overline{\theta}' - \frac{2\alpha}{t-c}[\theta] - 2\overline{\theta}[\theta] - \frac{\alpha^2}{(t-c)^2} - \frac{2\alpha}{t-c}\overline{\theta} - \overline{\theta}^2 + 2\overline{r}[\sqrt{r}] + \overline{r}^2$$

Both $[\theta]$ and $[\sqrt{r}]_c$ depend only on $1/t^i$ for i from 2 to ν. Note that the right hand side of the above equation does not have any terms that depend on $1/t^i$ for i from $\nu+2$ to 2ν. Therefore, the left hand side must equal 0. Therefore $[\theta] = \pm[\sqrt{r}]_c$.

Next, the right-hand-side coefficient of $1/t^{\nu+1}$ is $\pm\nu a \mp 2\alpha a + b$ where a is the coefficient of $1/t^\nu$ in $[\sqrt{r}]_c$ and b is the coefficient of $1/t^{\nu+1}$ in $r - [\sqrt{r}]_c^2$. The left-hand-side coefficient of $1/t^{\nu+1}$ is 0, so one can conclude that $\alpha = 1/2(\pm b/a + \nu)$.

Therefore, if c is a pole of r with order $2\nu \geq 4$, then the component of the partial fraction expansion of θ at c is:

$$\pm[\sqrt{r}]_c + \frac{\alpha}{t-c} \text{ where} \qquad \alpha = \frac{1}{2}\left(\pm\frac{b}{a} + \nu\right)$$

If r doesn't have a pole at c, then there still could be a pole in θ, but it must have the following properties. First, $[\theta] = 0$ and $-\alpha + \alpha^2 = 0$ as in the first order pole case. However, now the solution $\alpha = 0$ is valid. Therefore, the partial fraction expansion of θ at c would be either 0 or $1/(t-c)$.

Next, collect the terms calculated thus far:

$$\theta = \sum_{c \in \Gamma} \left(s(c)[\sqrt{r}]_c + \frac{\alpha_c}{t-c} \right) + \sum_{i=1}^{d} \frac{1}{t-d_i} + R$$

Note that $s(c) = \pm 1$ depending on which α is chosen (+ or -). The d singularities at d_i are the singularities of θ that are regular points of r. These are not determined yet. These will be computed last, as they represent the P'/P term mentioned above.

The next step will be to compute $R \in \mathbb{C}[t]$. To do so, the expansion of r and θ at

$t = \inf$ will be considered in the same way as above. Consider:

$$\theta = R + \frac{\alpha_{\inf}}{t} + \mathcal{O}(\frac{1}{t^2})$$

There are three cases to consider:

If r has order $\nu > 2$ at inf, then $r = \mathcal{O}(\frac{1}{t^\nu})$. Therefore, the Riccati equation becomes (for $1/t^2$ terms): $-\alpha_{\inf} + \alpha_{\inf}^2 = 0$. Therefore $\alpha_{\inf} = 0$ or 1.

If r has order 2 at inf, then $r = \frac{b}{t^2} + \mathcal{O}(\frac{1}{t^3})$, so the Riccati equation becomes: $-\alpha_{\inf} + \alpha_{\inf}^2 = b$. This implies that $\alpha_{\inf} = \frac{1}{2} \pm \frac{1}{3}\sqrt{1+4b}$.

If r has order $\nu < 2$ at inf, then first recognize that by the necessary conditions ν is even. The argument is similar to the above argument for finite poles of higher order. The same methodology finds

$$R = \pm[\sqrt{r}]_{\inf},\ \alpha_{\inf} = \frac{1}{2}\left(\pm\frac{b}{a} - \nu\right)$$

where -2ν is the order of r at inf, a is the leading coefficient (ν'th) of $[\sqrt{r}]_{\inf}$, and b is the $1/t^{\nu+1}$ coefficient of $r - [\sqrt{r}]_{\inf}^2$.

Therefore, it is now known that θ has the form

$$\theta = s(\inf)[\sqrt{r}]_{\inf} + \sum_{c\in\Gamma}\left(s(c)[\sqrt{r}]_c + \frac{\alpha_c}{t-c}\right) + \sum_{i=1}^{d}\frac{1}{t-d_i}$$

Next, one recognizes that the coefficient of $1/t$ in the Laurent series expansion of θ at inf is $\alpha_{\inf}$. This must match the order of the terms in the equation, meaning $\alpha_{\inf} = d + \sum_{c\in\Gamma}\alpha_c$. Therefore, one can find d

$$d = \alpha_{\inf} - \sum_{c\in\Gamma}\alpha_c \in \mathbb{N}$$

Note that if there is no $d \in \mathbb{N}$ that satisfies this equation, then the set of α_c calculated will not work. If there is no set of α_c such that such a d exists, then there is no closed-form solution for case 1.

To find the degree d polynomial P and complete the algorithm, remember that $\theta = \omega + P'/P$. Define ω

$$\theta = s(\inf)[\sqrt{r}]_{\inf} + \sum_{c \in \Gamma} \left(s(c)[\sqrt{r}]_c + \frac{\alpha_c}{t - c} \right)$$

and

$$P = \prod_{i=1}^{d} (t - d_i)$$

Plugging θ into the Riccati equation again, one obtains:

$$P'' + 2\omega P' + (\omega' + \omega^2 - r)P = 0$$

Since P is defined as a d degree polynomial, one can use the method of undetermined coefficients to solve for P. If a P is identified, then $\eta = Pe^{\int \omega} = e^{\int \omega + P'/P}$ is a solution of the differential equation since $\theta = \omega + P'/P$ is a solution of the Riccati equation. This completes case 1 of the Kovacic algorithm.

Similar approaches are used for the other two cases of the Kovacic algorithm. The primary differences are that instead of the Riccati equation, other differential equations are used based on the invariant of the Galois group according to that particular case. This is used to construct a value similar to θ observed above. Finally, there is a more complex relationship between η and the new 'θ' value than above. That having been said, the approach and intuition is identical to the above case.

6.3 Kovacic Algorithm Details

This work will use the Kovacic algorithm as a foundation to create an expansion algorithm with the property that its "space of functions" (the set of functions that can be returned by the expansion algorithm) is very large – the set of all possible functions that are exact solutions to second order differential equations with coefficients in $\mathbb{C}(t)$.

At a high level, the expansion algorithm uses the Kovacic algorithm to translate

the original ODE in terms of $y(t)$ into a second, different ODE in terms of a different dependent variable: $a_{n-1}(t)$ (the reason for the choice of this variable name is given later in this section). This second ODE in terms of $a_{n-1}(t)$ has the property that if the first ODE (in terms of $y(t)$) has a closed-form solution, the second ODE has a solution where $a_{n-1}(t)$ is a rational polynomial in t.

Consider the case this work is focused on – where the first ODE in terms of $y(t)$ has no closed-form solution. The corresponding $a_{n-1}(t)$ is therefore not in $\mathbb{C}(t)$. However, one can use order (p, q) Padé expansion on $a_{n-1}(t)$ to compute some $\hat{a}_{n-1}(t)$. By definition, $\hat{a}_{n-1}(t)$ is a fractional polynomial that matches $a_{n-1}(t)$ to order $p+q$. Because of the direct relationship between $y(t)$ and $a_{n-1}(t)$, the $\hat{a}_{n-1}(t)$ can then be used to construct an approximate $\hat{y}(t)$, where $\hat{y}(t)$ is a closed-form solution to *some* second order ODE that approximates the original ODE. This is the essence of the expansion described in Section 9.4 in a nutshell.

This section explores the relationship of the first ODE and its dependent variable $y(t)$ to the second, transformed ODE and its dependent variable $a_{n-1}(t)$. In particular, the Kovacic algorithm is used to derive the second ODE from the first ODE.

The Kovacic algorithm [90] identifies closed form solutions of second order linear differential equations with coefficients in $\mathbb{C}(t)$:

$$a(t)y''(t) + b(t)y'(t) + c(t)y(t) = 0$$
$$a(t),\ b(t),\ c(t) \in \mathbb{C}(t)$$

Note that equations of the above form can, without loss of generality, be written as $y''(t) = r(t)y(t)$ with $r(t) \in \mathbb{C}(t)$ by transforming the independent variable $y(t) = y_1(t) \exp\left(-\int \frac{b}{2a} dx\right)$ as discussed in the previous section.

This algorithm represents a culmination of study of differential Galois theory in that it enumerates all possible forms of a function that is a solution to a second order differential equation with coefficients in $\mathbb{C}(t)$, and uses this enumeration to "match" the correct solution. This algorithm rests on Theorem 6.2.1 [55].

Before continuing, recognize that Theorem 6.2.1 enumerates the space of possible

functions that can be exact solutions to second order linear ODE's with coefficients in $\mathbb{C}(t)$. These solutions will be of the form $\exp\left(\int \omega\right)$, where $\omega \in \mathbb{C}(t)$ or ω is algebraic of degree 2, 4, 6, or 12 over $\mathbb{C}(t)$. Note that 12 is the highest algebraic degree that ω can have. Define this class of functions to be "Kovacic closed-form":

Definition 6.3.1. *A function $f(t)$ is "Kovacic closed-form" ($f(t) \in K$) if and only if it is contained within the class of functions described in cases 1-3 of Theorem 6.2.1:*

1. $y(t) = e^{\int \omega}$, *where* $\omega \in \mathbb{C}(t)$.
2. $y(t) = e^{\int \omega}$, *where* ω *is algebraic of degree 2 over* $\mathbb{C}(t)$.
3. $y(t) = e^{\int \omega}$, *where* ω *is algebraic of degree 4, 6, or 12 over* $\mathbb{C}(t)$.

The final step is to use the ODE and ω to construct a second differential equation, whose solution must be a rational polynomial if $y(t)$ is a Kovacic closed-form function.

This final step is achieved by plugging the solution form in Definition 6.3.1 into the second order linear ODE. This results in a differential equation (a Riccati equation) involving ω as the independent variable. The requirement that ω be algebraic of degree n over $\mathbb{C}(t)$ results in Proposition 6.3.2 (See Proposition 3 in [55]):

Proposition 6.3.2. *For a differential equation $y''(t) = r(t)y(t)$ whose solution is of the form $y(t) = e^{\int \omega}$, where ω is algebraic of degree n over $\mathbb{C}(t)$, it is true that $y(t)$ is a solution to the differential equation if and only if ω is one of the roots of*

$$A(\omega, \overline{a}) = \omega^n - \sum_{i=0}^{n-1} \frac{a_i(t)}{(n-i)!}\omega^i \tag{6.1}$$

with the coefficients $\overline{a} = (a_{n-1}, \ldots, a_1, a_0)$ in some differential extension of $\mathbb{C}(t)$ de-

fined by:

$$\begin{aligned} a_n &= -1 \\ &\vdots = \vdots \\ a_{i-1} &= -a_i' - a_{n-1}a_i - (i+1)(n-i)a_{i+1}r(t) \\ &\vdots = \vdots \\ a_{-1} &= 0 \end{aligned}$$

The critical part of Proposition 6.3.2 (not shown) is that one can combine the equations for each a_i into a single order-n ODE whose independent variable is $a_{n-1}(t)$. By the construction of Proposition 6.3.2, if the first ODE (in terms of $y(t)$) has a solution of the form in Definition 6.3.1, then $a_{n-1}(t) \in \mathbb{C}(t)$. Note that the $a_{n-1}(t)$ term in Proposition 6.3.2 is the $a_{n-1}(t)$ term mentioned at the beginning of this section.

After deriving this second differential equation in terms of $a_{n-1}(t)$, the *exact* Kovacic algorithm uses Laurent expansions at the singular points of $r(t)$ to solve for a_{n-1}. This results in a solution that is in $\mathbb{C}(t)$ if it exists. However, if there does not exist an exact closed-form solution to the differential equation, the algorithm simply fails – there is no $a_{n-1}(t) \in \mathbb{C}(t)$ that solves the ODE.

However, as discussed at the beginning of this section, if the original ODE (in terms of $y(t)$) does *not* have a closed-form solution, then one can use Padé expansion on $a_{n-1}(t)$ to calculate an approximate $\hat{a}_{n-1}(t)$. Because Padé expansion by definition results in a fractional polynomial, $\hat{a}_{n-1}(t)$ corresponds to some $\hat{y}(t)$ that approximates $y(t)$, where $\hat{y}(t)$ is a Kovacic closed-form function as in Definition 6.3.1.

In conclusion, the Kovacic algorithm provides a one-to-one relationship between the Kovacic closed-form functions and the set of fractional polynomials. As a result, using Padé expansion as a foundation, one can construct a new expansion in terms of the Kovacic closed-form functions, a superset of the fractional polynomials.

One may now define the "Kovacic expansion" discussed further in Section 9.4.

Definition 6.3.3. *Define a "Kovacic expansion" of $y(t)$ at t_0 of order k to be a function that, for a given n (as defined in Proposition 6.3.2), is a (p, q) Padé expansion for $a_{n-1}(t_0)$, where $p+q = k+n-2$, and $a_{n-1}(t)$ is related to $y(t)$ as in Proposition 6.3.2.*

It is not yet clear why p, q, k, and n have the relationship stated in Definition 6.3.3. Informally, one must add $n-2$ to k to compensate for additional derivatives in the equation for a_{n-1} at higher values of n. This is shown in Lemma 9.5.1. Using Definition 6.3.3, define the set of functions K_k:

Definition 6.3.4. *The set of of functions that are Kovacic expansions (Definition 6.3.3) of order k are denoted K_k.*

One final item of note is that the Kovacic algorithm has five possible cases: $n = 1, 2, 4, 6, 12$. In the case where one is searching for exact solutions, one can eliminate one or more of these cases with sufficient conditions. However, when there is no exact solution and one is performing an expansion, there is no immediate reason why one value of n should be preferable to the other. Therefore, one must consider all five cases and compare the convergence properties of the Padé expansion of each $a_{n-1}(t)$.

7 - Circuit Complexity of Analog Computing

Chapter 5 introduces three informal criteria for PPUF systems. However, the use of complexity theory to formalize these criteria is non-trivial, as the PPUF hardware in general is an analog physical system, obeying some set of ordinary differential equations.

In order to proceed, recognize the PPUF hardware as an analog computer, and the problem may be restated as comparing the complexity of analog computing versus digital computing. This much more broad problem has implications well beyond use in Public Model Physical Unclonable Functions. Therefore, this chapter and Chapter 8 proceed with dual purposes. I provide a theory supporting the security of a formal PPUF construction. However, I also provide criteria for the speedup of more general analog computing systems, and provide more intuitive description and justification for the potential advantages of analog computing over digital computing for some specific computational problems.

7.1 Introduction to Analog Computation

Although analog computation has been in use far longer than digital computation, digital computation has become ubiquitous in the modern world, while analog computation is effectively nonexistent. The reasons for the superiority of digital computation are well understood [141, 152]. Noise tolerance, scalability, and abstraction are at the center of this advantage.

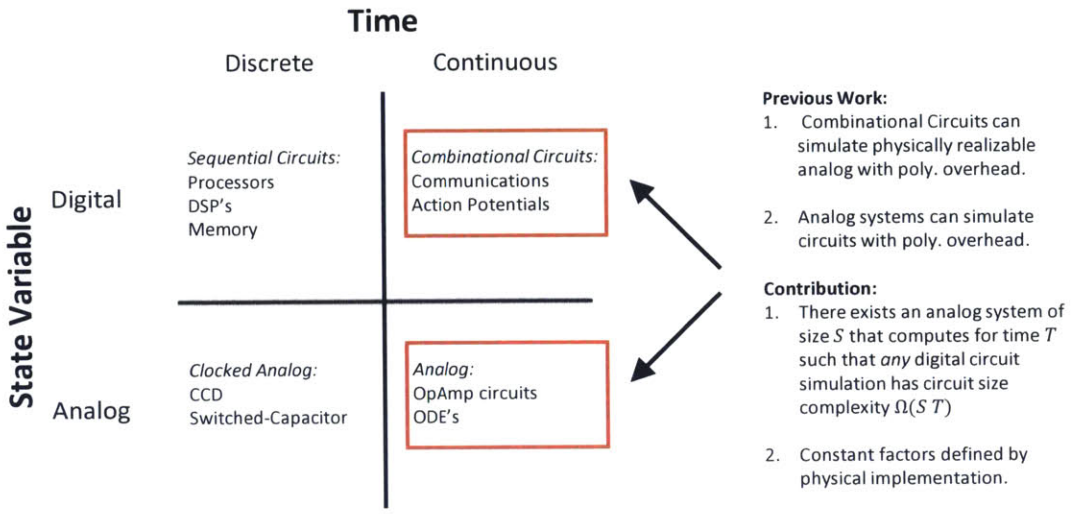


Figure 7-1: A depiction of the relationship of analog, digital, and analog/digital hybrid computational models. This work focuses on the complexity theoretic relationship between continuous-variable continuous-time analog systems, and discrete-variable continuous-time combinational circuits.

However, recent years have seen a resurgence of interest in analog computation – in particular as an alternative to the existing digital computation model for certain applications – e.g., machine learning [111, 24, 152, 122, 45]. Clearly, there is belief in the community that there is an advantage to be gained by using analog computation for certain problems. This belief stems from the empirical observation that large analog systems with fast dynamics are typically impossible to simulate in real time, even if the system in question is both purely classical and non-chaotic [119]. However, quantification and theoretical justification of such a belief has eluded the scientific community for almost a century.

This work provides a formal conjecture (Conjecture 8.4.1), and an associated construction wherein an analog computer has a constant-factor advantage over a digital computer that computes the same function. It provides a theoretical argument for *why* analog computation is more powerful than digital in certain cases, and for *which* types of problems analog computation can exhibit this advantage. In particular, this work analyzes the circuit complexity of specific analog computation models, resulting in a potential constant-factor circuit size complexity advantage of Shannon's General Purpose Analog Computer (GPAC) [147] over the circuit size complexity of *any* digi-

tal simulator. It is shown that there exists a GPAC device with size S that computes for time $t_f - t_s$, where any digital simulator has circuit size complexity $\Omega(S(t_f - t_s))$.

This result uses a singular hardness assumption (Conjecture 8.4.1) regarding numerical integration of differential equations that is discussed and justified in Chapter 9. This result is depicted in the context of analog/digital computational models in Figure 7-1.

7.1.1 Description of Main Result

Using a single conjecture, this work provides a theoretical argument that certain analog systems can have a circuit size advantage over any digital simulation, and therefore that there exists an advantage to using analog over digital for certain computational problems.

Note that the difference in computational power between analog and digital models of computation is not an asymptotic separation – the analog system has an advantage of at most a *constant factor* improvement over digital simulation.

The constant factors relating the efficiency of these analog and digital systems depend on implementation.

7.1.2 Related Work: Analog Computation and Complexity Theory

While there is a clear, generally accepted model of digital computation and computational equivalency – the Turing machine and Turing machine equivalency, there exists no such analog computational model with equivalent acceptance by the community [24]. Indeed, there are a multitude of analog and analog/digital hybrid computational models that have been proposed over decades [147, 148, 20, 141]. These fall into the four broad categories of computational models shown in Figure 7-1.

This work focuses on *continuous* analog computation, as opposed to "clocked" analog systems with continuous state variables under discrete time (the lower-left corner of Figure 7-1). There are two reasons for this.

- Clocked analog systems typically require the analog component to achieve steady state before the next clock cycle. In essence, the dynamics of the analog system are ignored except for the speed at which the system relaxes. This work uses the computation time $t_f - t_s$ as a parameter of the problem, which allows asymptotic analysis of the circuit size complexity as $t_f - t_s$ increases.

- Second, clocked analog systems are (in most cases) already easily compared with digital systems since the analog operation performed during a single clock cycle is much simpler, and typically comparable to an equivalent digital circuit. For example, in [152], the analog component performs multiply, accumulate, and compare functionality. There is a clear comparison between this numeric computation and a digital counterpart, since these functions are arithmetic, map well onto digital logic, and are well studied. Continuous analog systems are much less straightforward, as they are typically highly complex non-arithmetic functions of both state and time.

Continuous analog computational models are believed (and in some cases proved) to be equivalent to (or subsumed by) the Turing model of digital computation [25, 129]. Indeed, this belief has led to an extended version of the Church-Turing thesis – informally that all reasonable classical models of computation can simulate each other with polynomial time overhead and constant-factor space overhead [168]. This extended version of the Church-Turing thesis applies to analog computational models with respect to digital simulators. Therefore, much of the research effort into analog computational models over the past decades has focused on confirming that the proposed models of analog computation do conform to the extended Church-Turing thesis [24].

Although a statement that certain analog computational models are equivalent (can be simulated in at most polynomial overhead) to Turing machines is useful for many reasons, it loses the precision required to study the difference in computational power between the analog computational model and a digital model. Intuitively, there are many analog systems that can be simulated with at most polynomial time

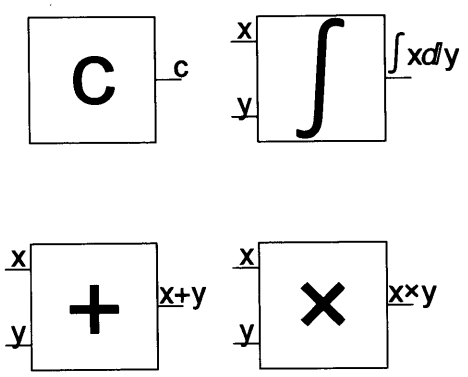


Figure 7-2: The different building blocks of any GPAC computer. Connection of these building blocks allows for the construction of analog computers capable of approximating ODE initial value problems (IVPs).

overhead that cannot be simulated in real-time as discussed above.

In the class of continuous analog computation, this work focuses on Shannon's general purpose analog computer (GPAC) model [147]. In essence, this model of analog computing follows the model of Bush's differential analyzer [32], and is a model that is tailored to solving initial value problems for ordinary differential equations. GPAC is generally described as any computer that is the composition of building blocks shown in Figure 7-2.

The theory of computational complexity of ordinary differential equations (analog ODE solvers represent one model for analog computation) has shown that the class of analytic ordinary differential equations (the class that is generally thought to be physically realizable, and thus solvable by an analog computer) can be simulated efficiently by polynomial time algorithms on a Turing machine [83, 84, 85, 129]. This is the extent of the granularity of analysis, so it does not capture the potential computational advantages of analog system over their digital counterparts.

One interpretation of the above setback is that Turing machine equivalency is not the correct objective for analog computation. It is reasonable to expect that modern digital processors will not (and should not) be replaced by analog processors, so it is not necessarily fruitful to pursue fully universal computation out of the alternate model. Instead, consider that analog computation should be seen as a hardware ac-

celerator (e.g., [152, 31]) solving a specific class of problems faster than a general purpose computer. This is not wholly consistent with the current thread of research, which focuses on establishing Turing equivalence or attempting to identify opportunities for "hypercomputation" – showing a class of problems that is asymptotically more efficient with analog computation [24].

This work proposes that continuous analog computers should be compared to combinational digital circuitry designed to perform an equivalent function. Therefore, this work proposes that circuit complexity is the correct model to provide comparisons between continuous analog (denoted simply as "analog" in the rest of this work) and digital computation.

7.1.3 Circuit Complexity and Analog Computation

This work will only consider a subset of analog computation – solving ordinary differential equation initial value problems (ODE IVPs). Furthermore, this work will only consider a subclass: the second-order differential equations described in Section 7.1.4.

The choice of second order ODE IVPs is made to simplify the analysis and enable the use of certain specific theoretical tools described in Section 7.1.4. The general structure of higher order ODEs and highly parallel coupled systems are less understood mathematically, so these theoretical tools do not exist at higher order.

Consider the impact of restricting to this subset of analog computation. First, an analog computer that simulates a second order ODE IVP from t_s to t_f has very little internal parallelism as the state consists entirely of the dependent variable and its first derivative. Therefore, size $S = \theta(1)$, and time $t_f - t_s$.

If there exists an upper bound on the maximum frequency component of the dependent variable (which is always the case for physically realizable systems), then a standard Runge-Kutta ODE integrator can integrate from t_s to t_f in $\theta(t_f - t_s)$ steps, as the minimum timestep is determined by this maximal frequency component. Runge-Kutta has $O(1)$ memory complexity, so the digital simulation is $\theta(t_f - t_s)$ circuit size complexity – asymptotically equivalent to the analog computer.

The challenge is to show that this worst case is also the best that any digital simulator can accomplish, and provide a lower bound on the constant factor for the digital simulator. Showing a constant-factor relationship between the analog and digital systems would entail finding a lower bound on the circuit depth. One can also show a constant factor circuit size relationship using fewer assumptions.

The relative constant factor is determined by the relative physical parameters between the analog and digital systems. Unfortunately, this fact strays somewhat from the mathematical abstraction up to this point. This work provides direction regarding how to establish this factor by comparing dynamic speed to basic CMOS building blocks (e.g., a multiplier) in Chapter 10.4.

7.1.4 Preliminaries on Ordinary Differential Equations

Since the objective of this work is to identify cases where analog computation of an ordinary differential equation may have a constant factor circuit size advantage over a digital simulator, consider first the strengths of both analog and digital computation individually.

Analog systems may have fast dynamics and cheap (in terms of energy, time, and space) calculation of certain continuous functions (a digital computer can only compute arithmetic functions [141, 164, 152, 45]). However, an analog ODE simulator (in the GPAC model) must necessarily evolve in time directly and cannot take advantage of mathematical structure of the ODE to take shortcuts. Since digital computers are fast for general purpose algorithms, a digital simulation can leverage this structure to be more efficient than its analog counterpart (e.g., see example from Section 5.8).

In general, there are two possible situations where digital computers might be able to obtain such an asymptotic speedup. First, if the ODE has a closed-form solution. Second, if one can quickly compute an approximation that describes the ODE's solution to sufficient precision over a large region.

The first requirement is addressed by the theory of closed-form solutions to ordinary differential equations, known as Differential Galois Theory [88, 149] (cf. Chapter 6).

This theory is a differential analogue to the traditional Galois theory using a differential algebra (introductions to which may be found in the texts [79, 81, 80, 128, 88]). In essence, differential Galois theory studies field extensions to a differential field generated by elements in the ring of differential operators (ODEs), while traditional Galois theory studies field extensions generated by elements in the ring of polynomials. This theory is presented in Chapter 6.

A key achievement of this theory has been the creation of the Kovacic Algorithm [90, 142, 55]. This algorithm, when given a differential equation of a certain type, either (a) returns the closed-form solution of the differential equation, or (b) returns that there is no closed-form solution. This algorithm therefore encapsulates the *entirety* of the structure of solutions to ODEs in the class to which it applies. This algorithm is now widely used, coming as a standard subroutine in symbolic ODE solvers in both Matlab and Mathematica [1, 2].

The Kovacic algorithm only works for a certain class of differential equations: second-order linear ODEs whose coefficients are in $\mathbb{C}(t)$, where t is the independent variable. This work accordingly considers this same class of ODEs as candidates to be solved by analog computers.

In this way, there is a clear distinction between those ODEs that have closed form solutions (and therefore whose digital simulators will perform asymptotically better than their analog counterparts), and those ODEs that do not have such closed-form solution.

The second requirement – that no approximate closed form solutions can be computed – is present because GPAC analog computers by definition compute *approximations* of the differential equation. Although analog computers such as the GPAC compute on continuous variables, they cannot be allowed to have infinite precision. Such systems have been shown to be able to solve NP complete problems, but are generally thought to be physically unrealizable [145, 4].

This problem of approximate closed-form solutions turns out to be a significant challenge and will be the focus of this work. By computing an upper bound on the size of the domain where these approximate closed-form solutions accurately model

the solution to the underlying ODE, this work computes a lower bound on the circuit size complexity required to model this ODE using any digital simulation.

ODE Approximation and Numerical Integration

In the case where an ODE does not have a closed-form solution (most ODEs do not have closed-form solutions), approximation methods must be used. In the case of initial value problems, numerical integration techniques are used to integrate the ODE from the initial conditions to some time later [47].

All such numerical integration techniques must contend with the inherent locality of ODE IVPs – all information regarding the state of the system is given at a single point in time, and the differential equation describes infinitesimal changes away from this point.

Therefore, local expansions of varying types are used in such numerical integration. Without loss of generality, a numerical integration technique follows a simple procedure to integrate a dependent variable y (which may be scalar or vector-valued) from $t = t_s$ to $t = t_f$ with initial conditions $y(t_s) = y_s$.

1. Set $t = t_s$, and set state: $y = y_s$.
2. Compute expansion of $y(\cdot)$ around t.
3. Compute approximate value of $y(t + dt)$. Update state. Set $t = t + dt$.
4. If $t = t_f$, Stop. Otherwise go to (2).

The expansion in step (2) may be a highly complex algorithm and require approximation of y at additional points (e.g., as in multi-step methods) [33]. An "expansion algorithm" is defined as in Section 8.2.

In the above algorithm, the performance of the expansion algorithm determines the performance of the integration routine. If an expansion algorithm can quickly find a closed-form solution that accurately describes $y(t)$ over a large domain, the integration routine performs very well.

Conversely, if one can show that any expansion algorithm of finite order will only be an accurate (to within ϵ, for example) model of the actual function inside of a relatively small finite domain, then one can bound the performance of the integration

routine.

The domain bounds how large dt can be in the above algorithm, which in turn results in a lower bound on the number of steps that need to be taken between $t = t_s$ and $t = t_f$. In Section 8.5, this concept is formally defined in terms of the expansion algorithm.

The critical step in the above analysis is how to bound the region where *any* expansion algorithm at a given order correctly approximates $y(t)$ to sufficient accuracy.

7.1.5 Circuit Complexity of ODE Approximation: Contributions

To bound the region of accuracy of this local expansion, first recognize that there are two factors that determine this region of accuracy: (1) expansion order, and (2) estimation of sub-dominant terms.[1]

This work then proposes a conjecture that all numerical integration algorithms use a local expansion of finite order (Conjecture 8.4.1).

If this is the case, then even with an arbitrary function, the best possible expansion of the function still has a bounded region of accuracy. This conjecture is shown to be supported by the last century of work in approximating solutions of differential equations.

Finally, this method is used to lower bound the number of expansions required to approximate a class of ODEs. It is recognized that this lower bound is also a lower bound on the circuit size complexity of any digital circuit that simulates the ODEs.

To build intuition for the implication of the above conjecture, I construct in Chapter 9 a new type of expansion using the Kovacic algorithm. It is shown that modern asymptotic approximations of second order linear ODEs are subsumed by the proposed expansion algorithm. This is due to the fact that the Kovacic expansion methodology has a much larger function space than all modern asymptotic expansion

[1] An expansion algorithm computes the first k terms. However, it can estimate higher order terms based on its knowledge of the form of the function that is being expanded. I.e., if the function being expanded is from a certain class of functions like the rational polynomials, then higher order terms of the expansion can be estimated based on this fact. See Section 9.3.

algorithms applicable to this class of ODEs.

To see how the space of functions used by the expansion affects the region of accuracy of the expansion, consider Taylor series of order k as compared to a Padé expansion of order (p, q) (p and q are the polynomial degrees in the numerator and denominator respectively). In the case where $p + q = k$, one empirically finds that for most functions and most choices of p, q, the Padé expansion will more accurately model the function over a wider domain. This widely recognized behavior drives many of the numerical convergence acceleration algorithms that are used [165].

The Padé expansion converges faster than Taylor series in general because it has a larger space of functions to draw from – a Taylor series uses polynomials, while Padé expansion uses fractions of polynomials.

The contribution can therefore be understood as increasing the space of functions as far as possible, and using it to identify the *best possible* expansion at a given order of a solution to a certain ODE. Instead of explicitly calculating the expansion, I show that even with perfect analysis of the subdominant terms, there is still uncertainty in the ODE solution, because the expansion is only to finite order. This method is used to bound the possible region where expansions of a given order can be accurate. This region is then used to lower bound the size of any digital circuit tasked with simulating the ODE.

8 - PPUF/Analog Computing Formalism and Theory

As discussed in the previous section, an asymptotic lower bound on PPUF simulation time is equivalent to an asymptotic lower bound on the circuit depth of any simulating hardware. Furthermore, I recognized that a PPUF was an instance of the more general problem of comparing digital and analog computing modalities. In this case, one is more interested in a simpler comparison between the circuit *size* of any simulating hardware versus a digital simulator.

In this chapter, I show an lower bound on PPUF simulation time if Conjecture 8.4.1 is true. Further, I show an lower bound on the circuit size complexity using the same assumption. The lower bound on circuit depth complexity is in the random oracle model, but the lower bound on circuit size complexity does not have this requirement.

Now, consider the circuit size of any simulator for an analog initial value problem (IVP) that evolves from time t_s to time t_f. Consider an IVP whose ODE is of the form $y''(t) = r(t)y(t)$ – the coefficient $r(t)$ must depend somehow on a set of problem parameters. Consider that these parameters affect the value of $r(t)$ at different points in time t. For the sake of example, consider that the effect of a single parameter on the value of $r(t)$ is temporally localized to some constant-sized region in time. In this model, it is reasonable to consider that this IVP has $\Theta(n)$ parameters overall. This is because of a previous, informal observation that IVPs are easy to simulate (have an asymptotic speedup when simulated with a general purpose processor) when they have reached steady state. If the parameterized regions of $r(t)$ are far enough apart,

the system reaches steady state in between.

With this in mind, there is already a trivial asymptotic lower bound of $\Omega(n)$ on the circuit size of any digital simulator that simulates the parameterized IVP, as there are $\Theta(n)$ inputs to the circuit!

However, this lower bound turns out not to be useful. Recall that the purpose of establishing a lower bound on circuit size complexity for analog computing, and circuit depth complexity for PPUF applications is to compare constant factors between analog computing and digital computing modalities. The above asymptotic analysis does not provide any insight as to this constant factor.

In particular, consider a case where one parameter of the IVP for some reason has little to no impact on the final outcome of the system. In this case, an optimal simulating circuit may not even use the parameters as a part of the larger computation or only use them for a very small part of the overall computation. Therefore, it is completely possible that the contributed circuit size for that particular IVP parameter may be very low (down to a few gates).

There is no formal way to reason about this contribution, and there is no lower bound on the constant factor in front of the $\Theta(n)$ term. Therefore, a different approach must be used.

Instead, I will use Conjecture 8.4.1 to show a lower bound (non-asymptotic) on the number of local expansions computed in order to perform numerical integration from t_s to t_f.

In order to accomplish this, a new formalism is introduced for analyzing differential equations in this way. Some preliminary results are provided that both justify the formalism and provide direction as to the types of IVPs that should be considered.

Therefore, this chapter follows the outline:

1. Define an "initial value problem" (IVP) in the context of second order, linear time varying differential equations.
2. Recognize that for a differential equation to be useful for computation, it must be "entropy-maintaining." Given initial conditions to the ODE IVP with some entropy, this entropy must be maintained throughout the evolution of the ODE

IVP. Simply put, the ODE IVP cannot "forget" the initial conditions. This puts restrictions on the type of ODE IVP to be considered.

3. Recognize that any physically implementable ODE IVP must be bounded. This has ramifications for the type of ODE operator that restrict the possible types of ODE IVPs to be considered.
4. Recognize an equivalence class of ODE IVPs as those that are translations/ rotations/magnifications of each other.
5. Recognize that ODE IVPs that are localized in time (all computation occurs within a certain interval of time) may be stitched together using the above equivalency to generate sequential computations.
6. Identify how to sample from ODE IVPs randomly. Argue generality based on the parameterization of the distribution.
7. Formalize definitions of IVP and expansion/numerical integration algorithm.
8. Make an supported conjecture regarding the form of any numerical integration problem that solves IVPs of the form defined earlier.
9. Statistically show an empirical observation regarding the entropy of IVP solutions in the presence of the computation of a single expansion.
10. Given a large ODE IVP that may be broken into n independent sub-problems, show that since each sub-problem is independent, the complexity of solving the overall system is $> n \times \min(\mathsf{Cost}(\mathsf{Exp}^{\mathsf{A}}{}_k)$ circuit size complexity, where $\mathsf{Exp}^{\mathsf{A}}{}_k$ is a family of expansion algorithms for an adversary A.
11. Use a similar argument in conjunction with a random oracle to use the solution of one IVP to derive the parameters for a second IVP. With this feed-forward functionality, show a circuit-depth lower bound for use in PPUF applications.
12. Link the above circuit complexity analysis with existing formalism for PUF applications to derive a complete, formal construction for a PPUF.

The notation for this section is summarized below:

- $h(Y)$: the differential entropy of a continuous random variable Y.
- **IVP**: Set of parameters defining an initial value problem. See Definition 8.1.4.
- **Rslt**: pair of $(t_f, y(t_f))$ that indicates the solution of the IVP at the final time

t_f.

- $C(x)$ (sometimes C): Curve of integration parameterized by some real value x. See Definition 8.1.5 for a "proper" curve.
- L: Differential operator. For this work, it will always be of the form $\partial_t^2 - r(t)$.
- $[L]$: Equivalence class of the differential operator L (See Definition 8.1.15).
- $r(t)$: ODE Coefficient. All ODEs in this section are of the form $y''(t) = r(t)y(t)$. See L, the differential operator above.
- t_s: starting point of numerical integration of an IVP.
- t_f: ending point of numerical integration of an IVP.
- R: radius of disc on $\mathbb{C}$ inside of which all poles/zeroes for an IVP are located. See Definition 8.2.2 and Figure 8-1.
- T: used to denote the distance between clusters of poles/zeroes (see Figure 8-2).
- **IC**: initial conditions of the initial value problem. Because the initial value problem is always linear, second order, **IC** is the pair $(y(t_s), y'(t_s))$.
- ϵ: Allowed error of the IVP. See Definition 8.2.1.
- δ: Error in encoding of analog variables for digital computation.
- k: Expansion order of an numerical integration algorithm.
- $(y,\ y')$: IVP state.
- M: PPUF Model represented by bitstring of length l.
- χ: Distribution of PPUF model over $\{0,1\}^l$.
- V: Challenge to PPUF/PAF. Represented as $\{v_i\}$ for i from 1 to n, where $v_i \in \{0,1\}^l$.
- $\mathcal{L}$: a language (See Definition 8.2.1).
- $U_{R,m,\delta}$: A family of initial value problems from Definition 8.2.2.
- $H_{R,m,\delta}$: A random oracle that returns elements from $U_{R,m,\delta}$.

8.1 Preliminaries

8.1.1 Differential Entropy

This work considers randomness of continuous variables, and as such requires the use of the differential entropy (Definition 8.1.1). See Chapter 9 of [40] for proofs and more extensive discussions of the following.

Definition 8.1.1. *Given some random variable X with continuous probability distribution function $f(x)$ with support set S (i.e., $f(x) > 0 \; \forall \; x \in S$), define the differential entropy $h(X)$:*

$$h(X) = -\int_S f(x) \log f(x) dx \tag{8.1}$$

Proposition 8.1.2. *Given a scalar-valued random variable with continuous probability distribution Y, and scalar a:*

$$h(aY) = h(Y) + \log |a|$$

Proposition 8.1.3. *Given a vector-valued random variable with continuous probability distribution $\mathbf{Y}$, and a matrix A:*

$$h(A\ \mathbf{Y}) = h(\mathbf{Y}) + \log |A|$$

Where $|A|$ is the absolute value of the determinant of A.

One should be aware of the significant differences between discrete entropy and differential entropy. Namely, discrete entropy is always greater than 0, while differential entropy does not have this requirement. Differential entropy also has units (i.e., changing the units of Y affects the differential entropy $h(Y)$).

Informally, knowledge to infinite precision of a variable corresponds to differential entropy approaching $-\infty$ (rather than 0, which is the case in discrete entropy).

8.1.2 Differential Equations

Definition 8.1.4 encapsulates the set of problems that this work studies: second order linear initial value problems.

Definition 8.1.4. *A linear second-order initial value problem* **IVP** *is the set* $\{L, C(x), t_s, \mathbf{IC}, \epsilon\}$*:*

- *The second order ODE operator:* $L(y(t), t) = y''(t) + r(t)y(t)$*. The corresponding ODE therefore can be written* $L(y(t), t) = 0$*. Let* $r(t) \in \mathbb{C}(t)$ *have poles at* p_i*, zeroes at* z_j*, and* $r_\infty = \lim_{t \to \infty} r(t)$*.*
- *The curve of integration* $C(x) \subset \mathbb{C}$ *as parameterized by variable* $x \in \mathbb{R}$*.*
- *The point of expansion* $t_s \in C(x)$*.*
- *The initial conditions* $\mathbf{IC} = \{y_0, y_0'\} \in \mathbb{C}^2$*.*
- *The maximum series truncation error* $\epsilon > 0$*.*

A "proper" integration curve is defined below. There are several intuitive properties that are encapsulated by Definition 8.1.5. For example, the curve must approach infinity as its parameter $|x|$ approaches infinity. This is because systems must always make *forward progress*, rather than re-treading the same state space or stopping in the middle of a computation.

Furthermore, setting $|C'(x)| > 0$ is justified because forward progress must never stop. Next, bounding $C''(x)$ prevents discontinuities in $y(C(x))$, which allows one to reason about the instantaneous behavior of the ODE. In addition, for physical realizability, require that $C(x)$ not traverse any poles of $r(t)$.

Finally, require analytic $C(x)$ in the limit of large $|x|$ because one must be able to reason about the asymptotic behavior of $C(x)$, $r(C(x))$, and $y(C(x))$ in this limit.

Definition 8.1.5. *A curve of integration* $C(x) \subset \mathbb{C}$ *is "proper" for a given* $r(t)$ *if it is continuous, can be parameterized by some value* $x \in \mathbb{R}$ *as* $C(x)$*, and has the following properties:*

- $C(x)$ *is analytic in the limit* $x \to \pm\infty$
- $\lim_{x \to -\infty} |C(x)| = \infty$*, and* $\lim_{x \to \infty} |C(x)| = \infty$

- $\lim_{x \to -\infty} C(x) \neq \lim_{x \to \infty} C(x)$
- $C'_{\text{MAX}} > |C'(x)| > 0 \ \forall x$
- *$C''(x)$ and $C'(x)$ are bounded for all x ($C'(x)$, $C(x)$ continuous)*
- *$C(x)$ does not go through any of the poles of $r(t)$*

Definition 8.1.6. *Define* **IVP** *with randomized initial conditions* **IC** *to be "Entropy maintaining" if:*

- *The integration curve $C(x)$ is proper.*
- *$h(\mathbf{IC}) = l$ with l bounded.*
- *$h(y(C(x))) = \Theta(lf(x))$, with $f(x) = \Theta(1)$.*

Proposition 8.1.7. *All IVPs as defined in Definition 8.1.4 are entropy maintaining.*

Proof. By Definition 8.1.4, the ODE operator is $L = y''(t) + r(t)y(t)$. Consider a broader class of ODEs[1] defined along the curve $t = C(x)$. Also, extend Definition 8.1.5 to include that $C(x)$ does not contain any poles of $b(t)$ or $a(t)$.

$$\frac{d^2}{dt^2}y(t) + a(t)\frac{d}{dt}y(t) + b(t)y(t) = 0$$

Assigning $t = C(x)$, $\hat{y}(x) = y(C(x))$, and $\hat{y}' = d/dx\ \hat{y}$ and using the chain rule, change the ODE to the form:

$$\hat{y}''(x) + \left(a(C(x))C'(x) - \frac{C''(x)}{C'(x)}\right)\hat{y}'(x) + b(C(x))C'(x)^2\hat{y}(x) \tag{8.2}$$

First, note $|C'(x)| > 0$ and $\lim_{x \to \pm\infty} |C(x)| = \infty$ in Definition 8.1.5, so the coefficients above do not have any finite or infinite poles when $x \in \mathbb{R}$. Therefore, the coefficients of Equation 8.2 are bounded for bounded x.

Consider the Forward Euler approximation of the matrix form of this ODE that converges to $\hat{y}(x)$ in the limit $\Delta x \to 0$.

[1]Note that ODEs of this class may be transformed by a change of coordinates into ODEs of the form $y''(t) = r(t)y(t)$.

$$\begin{pmatrix} \hat{y}((n+1)\Delta x) \\ \hat{y}'((n+1)\Delta x) \end{pmatrix} =$$
$$\begin{pmatrix} 1 & \Delta x \\ b(C(n\Delta x))C'(n\Delta x)^2\Delta x & 1+\left(a(C(n\Delta x))C'(n\Delta x) - \frac{C''(n\Delta x)}{C'(n\Delta x)}\right)\Delta x \end{pmatrix} \cdot \begin{pmatrix} \hat{y}(n\Delta x) \\ \hat{y}'(n\Delta x) \end{pmatrix} \tag{8.3}$$

By Proposition 8.1.3, one may consider the effect of n iterations of Equation 8.3 on the entropy of $\hat{y}, \hat{y}'$.

$$h\begin{pmatrix} \hat{y}(n\Delta x) \\ \hat{y}'(n\Delta x) \end{pmatrix} = h\begin{pmatrix} \hat{y}(0) \\ \hat{y}'(0) \end{pmatrix} + \sum_{i=0}^{n} \log \left| \begin{matrix} 1 & \Delta x \\ b(C(i\Delta x))C'(i\Delta x)^2\Delta x & 1+\left(a(C(i\Delta x))C'(i\Delta x) - \frac{C''(i\Delta x)}{C'(i\Delta x)}\right)\Delta x \end{matrix} \right| \tag{8.4}$$

Expand the determinant:

$$\begin{aligned} D(i) &= \left| \begin{matrix} 1 & \Delta x \\ b(C(i\Delta x))C'(i\Delta x)^2\Delta x & 1+\left(a(C(i\Delta x))C'(i\Delta x) - \frac{C''(i\Delta x)}{C'(i\Delta x)}\right)\Delta x \end{matrix} \right| \\ &= 1+\left(a(C(i\Delta x))C'(i\Delta x) - \frac{C''(i\Delta x)}{C'(i\Delta x)}\right)\Delta x - b(C(i\Delta x))C'(i\Delta x)^2\Delta x^2 \end{aligned}$$

Next, take a series expansion of logarithm in Equation 8.4 around $\Delta x \to 0$:

$$h\begin{pmatrix} \hat{y}(n\Delta x) \\ \hat{y}'(n\Delta x) \end{pmatrix} = h\begin{pmatrix} \hat{y}(0) \\ \hat{y}'(0) \end{pmatrix} + \sum_{i=0}^{n}\left(a(C(i\Delta x))C'(i\Delta x) - \frac{C''(i\Delta x)}{C'(i\Delta x)}\right)\Delta x + O(\Delta x^2) \tag{8.5}$$

Finally, recognize that for any finite interval of interest $n = \Theta(1/\Delta t)$ and taking the limit of $\Delta t \to 0$ and large n implies that the sum becomes an integral over the

curve of integration of the ODE. Consider that $X = n\Delta x$.

$$h\begin{pmatrix} \hat{y}(X) \\ \hat{y}'(X) \end{pmatrix} = h\begin{pmatrix} \hat{y}(0) \\ \hat{y}'(0) \end{pmatrix} + \int_0^X \left(a(C(x))C'(x) - \frac{C''(x)}{C'(x)} \right) dx + \lim_{\Delta x \to 0} \sum_{i=0}^{n} O(n\Delta x^2) \tag{8.6}$$

For terms that are $O(n\Delta x^2)$, recognize that this sum becomes $\lim_{\Delta x \to 0} O(X\Delta x)$. Since all of the coefficients of the ODE (Equation 8.2) are bounded, the determinant $D(i)$ is also bounded. Therefore, the coefficients at higher orders of Δx must also be bounded in Equations 8.5, 8.6. Therefore, since the interval $[0, X]$ is finite, the limit $\lim_{\Delta x \to 0} O(X\Delta x) = 0$, and the sum of $O(n\Delta x^2)$ terms in Equation 8.6 goes to 0.

The first integral must be asymptotically constant in X for the overall ODE to be entropy-maintaining. First note that for all IVPs matching Definition 8.1.4, $a(t) = 0$, so the integral reduces to:

$$\int_0^X \frac{C''(x)}{C'(x)} dx = \ln C'(X) - \ln C'(0)$$

Therefore, by this result and Definition 8.1.5, **IVP** is entropy maintaining since $|C'(X)|$ is bounded from above for all X by C'_{MAX}.

□

The entropy-maintaining property states informally that an IVP with variable initial conditions must have a system state that has significant dependence on these initial conditions over the curve of integration of interest. This is important, but not sufficient. The IVP must not have unbounded behavior on the curve of integration of interest. Therefore, the notion of boundedness is introduced.

This concept is required for several reasons:

- Most reasonable physical realizations of analog systems do not allow for unbounded internal state (e.g., a spring cannot stretch to infinity). In such cases, the system usually exhibits nonlinear higher-order behavior as the internal state variables become large (due to the fact that the linear system is an approxima-

tion of the overall dynamics of the system around some single point of operation).

- Measurement of such a physical system becomes very difficult if the state variable has too large of a range.

Definition 8.1.8. *An entropy-maintaining IVP is "bounded" if bounded* **IC** *implies bounded* $y(t)$, $y'(t)$ *for all* $t \in C(x)$.

Proposition 8.1.9. *There exist IVPs that are entropy maintaining and bounded:*

Proof. Consider **IVP** with:

- $L(y(t), t) = y''(t) + y(t)$, that is $r(t) = 1$.
- $C(x) = x$.
- $t_s = 0$.
- **IC** is random under a continuous probability distribution with differential entropy l.
- ϵ such that $\log(\epsilon) \ll l$.

First, $C(x)$ is trivially proper.

Second, by Proposition 8.1.7, it is entropy-maintaining.

Finally, given **IC** $= \{A, B\}$, where A, B are bounded, the general solution to $L(y(t), t) = 0$ is:

$$y(C(x)) = A\cos(x) + B\sin(x)$$

Therefore, A, B bounded implies $y(t)$ bounded for all $t \in C(x)$.

□

Now that the notion of entropy-maintaining and boundedness have been established, the requirements for any IVP that has both of these properties may be derived. This is important to the study, because any physical system hoping to have a computational advantage over CMOS must have both of these properties, as discussed above.

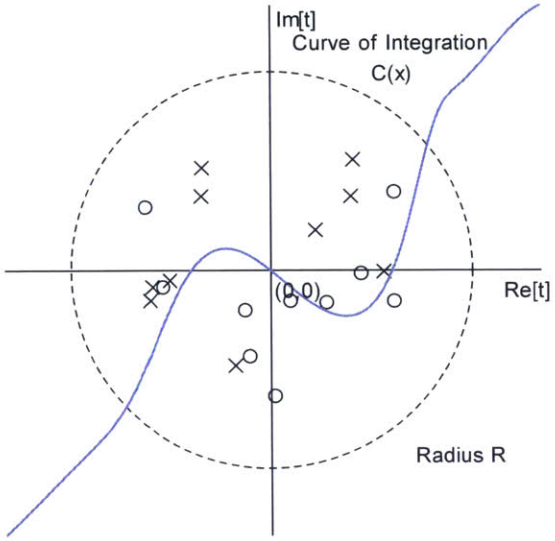


Figure 8-1: Figure of an IVP that follows Lemmas 8.1.12, 8.1.13. The poles and zeroes of $r(t)$ are marked, and the curve of integration is labeled. R is chosen such that all poles and zeroes are within radius R from the origin (which is in $C(x)$). Note that there are equal numbers of poles and zeroes, and $C(x)$ is asymptotic to infinity in opposite directions as $|x| \to \infty$. The asymptotic behavior of $C(x)$ must be chosen according to Lemma 8.1.12.

The following conditions are necessary and sufficient for an IVP to be entropy-maintaining and bounded.

- $\lim_{x \to \pm\infty} y(C(x))$ does not exist (DNE). $y(C(x))$ is bounded and approaches a limit cycle.
- $r(t)$ has a finite, equal number of poles and zeroes.
- If $\lim_{t \to \infty} r(t) = r_\infty \exp(i\phi)$, for $r_\infty \in \mathbb{R}$, then $\lim x \to \pm\infty C(x) = \pm \lim_{R \to \infty} R \exp(i(\phi/2 - \pi/2))$. This is depicted in Figure 8-1.

Note that for this work, distinguish between a limit that is infinite, and a limit that is finite, but does not exist (DNE).

Lemma 8.1.10. *If* $\mathbf{IVP} = \{L, C(x), t_s, \mathbf{IC}, \epsilon\}$ *is entropy-maintaining and bounded, then the following is true:*

$\lim_{x\to\infty} y_2(C(x))$	$\lim_{x\to\infty} y_1(C(x))$			
	0	Finite$\neq 0$	Infinite	DNE (bounded)
0	(2)	(3)	(1)	(4)
Finite$\neq 0$	(3)	(3)	(1)	(3)
Infinite	(1)	(1)	(1)	(1)
DNE (bounded)	(4)	(3)	(1)	

Table 8.1: Possible cases for limiting values of $y_1(C(x))$, $y_2(C(x))$. Exclusions are given as index numbers, referred to in this proof. The only possible case is that the limits $\lim_{x\to\infty} y_1(C(x))$ and $\lim_{x\to\infty} y_2(C(x))$ do not exist (DNE).

Let $y_1(t)$, $y_2(t)$ be linearly independent solutions of the ODE L. Then, neither $\lim_{x\to\pm\infty} y_1(C(x))$ nor $\lim_{x\to\pm\infty} y_2(C(x))$ exist (i.e., $y_1(C(x))$, $y_2(C(x))$ both oscillate as $|x| \to \infty$).

Proof. Let $y_1(t)$ be a solution of $L(y(t), t) = 0$, where $L \in$ **IVP**. Therefore, one knows for an ODE of the form $ay'' + by' + cy = 0$ (for **IVP**, $a = 1$, $b = 0$, $c = r(t)$), a second, linearly independent solution can be found in terms of the first (all variables are functions of t) [151]:

$$y_2 = y_1 \int \frac{\exp(-\int b/a \, dt)}{y_1^2} dt = y_1 \int \frac{1}{y_1^2} dt \tag{8.7}$$

Any set of initial conditions **IC** corresponds to a solution that is a linear combination of $y_1(t)$, $y_2(t)$. Therefore, consider Table 8.1 as the possible limits for $y_1(t)$, $y_2(t)$ on the curve of integration.

All but one of the outcomes in Table 8.1 is impossible. There are four observations that are required, indexed in the table to match the list below:

1. Since **IVP** is by definition bounded, neither the limits of $y_1(C(x))$ nor $y_2(C(x))$ can go to infinity.

2. Since **IVP** is entropy-maintaining, it cannot be the case that both limits go to zero.

3. Assume that $\lim_{x\to\infty} y_1(C(x))$ exists and is some finite value C_0. One may therefore construct a series expansion of $y_1(C(x))$ at $x \to \infty$ (note that this is possible

because $C(x)$ is proper and therefore is analytic in this limit, and $y_1(\cdot)$, $y_2(\cdot)$ are both meromorphic).

$$y_1(C(x)) = \sum_{i=0}^{\infty} C_i x^{-i} \tag{8.8}$$

Plug Equation 8.8 into Equation 8.7, and recognize that for the ODE $L \in \mathbf{IVP}$, $a = 1$, $b = 0$, and $c = r(C(x))$.

This gives the following expansion for $y_2(C(x))$ at large x.

$$y_2(C(x)) = \left(C_0 + \frac{C_1}{x} + O\left(\frac{1}{x^2}\right)\right) \int \frac{1}{C_0^2} - \frac{2C_1}{C_0^3 x} + O\left(\frac{1}{x^2}\right) \, dx \tag{8.9}$$

In Equation 8.9, $C_0 \neq 0$. Therefore, the integral diverges as $x \to \infty$. Therefore, $y_2(C(x))$ diverges at $x \to \infty$. Therefore, if $\lim_{x\to\infty} y_1(C(x)) = C_0$, where C_0 is finite, then $\lim_{x\to\infty} y_2(C(x))$ is unbounded.

The result is similar if one exchanges $y_1(\cdot)$, $y_2(\cdot)$.

4. Finally, consider that $\lim_{x\to\infty} y_1(C(x)) = 0$. In this case, consider that to leading order, $y_1(C(x)) = \Theta(t^k)$, for $k < 0 \in \mathbb{R}$. Then, by Equation 8.7, to leading order $y_2(C(x)) = \Theta(t^{-k+1})$. This is unbounded, therefore, it cannot be the case that $y_1(C(x))$ goes to 0, while $y_2(C(x))$ is bounded.

Therefore, it must be the case that neither $\lim_{x\to\infty} y_1(C(x))$ nor $\lim_{x\to\infty} y_2(C(x))$ exist.

□

Lemma 8.1.11. *If* $\mathbf{IVP} = \{L, C(x), t_s, \mathbf{IC}, \epsilon\}$ *is entropy-maintaining with some randomized* **IC** *with finite, non-zero* $h(\mathbf{IC})$*, then the following is true:*

$r(t) \in \mathbb{C}(t)$ *in the ODE operator* L *has no poles or zeroes at* $t \to \infty$.

Proof. First consider if $L = y'' + r(t)y$ has one or more zero at infinity $\left(\lim_{t\to\infty} r(t) = 0\right)$. Therefore, since $y(t)$ must be bounded along some proper curve of integration $C(x)$, $\lim_{x\to\pm\infty} y''(C(x)) = 0$ since $\lim_{x\to\pm\infty} r(C(x) = 0$. Therefore, $\lim_{x\to\pm\infty} y'(C(x)) =$ constant. Once again, for $y(t)$ to be bounded, this constant must be 0. This corresponds to $\lim_{x\to\pm\infty} y(C(x)) =$ constant. By Lemma 8.1.10, this cannot happen. Therefore, there are not one or more zeroes at infinity.

Next, consider if L has $k > 0$ poles at infinity $\left(\lim_{t\to\infty} r(t) = \infty\right)$. Therefore, let $r(t)$ to have a series expansion at $t \to \infty$:

$$r(t) = C_k t^k + O(t^{k-1}) \tag{8.10}$$

In this case, one may use WKB (see Chapters 11-13 of [170]) to construct an asymptotic expansion of $y(t)$ in the limit of large t. The WKB approach holds in the limit where $|\frac{dr}{dt} r^{-3/2}| \ll 1$ [170]. This criterion holds for any choice of k, C_i above.

The WKB expansion (to first order) for $r(t)$ is:

$$y(t) \approx y_{\pm,\text{WKB}} = r(t)^{-1/4} \exp\left(\pm i \int r(t)^{1/2} dt\right) \tag{8.11}$$

Note that this gives two linearly independent approximate solutions for the ODE $L \in \mathbf{IVP}$. Consider how these solutions behave with respect to any proper curve of integration $C(x)$. For $C(x)$ to be proper, it must be the case that $\lim_{x\to\infty} |C(x)| = \infty$.

Therefore, consider the limit of each $y(t)$ as t goes to infinity with some phase in the complex plane (i.e., let $y(x) = R(x)\exp(i\theta(x))$ as $x \to \infty$). First, recognize by plugging Equation 8.10 into Equation 8.11:

$$y_{\pm,\text{WKB}} \approx \left(C_k t^k + O(t^{k-1})\right)^{-1/4} \exp\left(\pm i \int (C_k t^k + O(t^{k-1}))^{1/2} dt\right) \tag{8.12}$$

Therefore, there are two possible outcomes for $\lim_{x\to\infty} y(R(x)\exp(i\theta(x)))$ depending on the choice of θ.

First, if $\arg(iC_k^{1/2} t^{k/2+1}) = \lim_{x\to\infty} \pi/2 + \arg(C_k)/2 + (k/2+1)\theta(x) = 0, \pm\pi, \ldots$, then the exponent in Equation 8.11 is purely imaginary (resulting in the $\exp(\cdot)$ term having magnitude 1). In this case, $\lim_{x\to\infty} y_{\pm,\text{WKB}}(R(x)\exp(i\theta(x))) = 0$, because of the coefficient in Equation 8.12: $r(t)^{-1/4}$. This is the case for both solutions, so one concludes that regardless of initial condition, $y(t)$ approaches 0, so the ODE cannot be entropy maintaining.

Second, if $\arg(iC_k^{1/2} t^{k/2+1}) \neq 0, \pm\pi, \ldots$, then the exponent in Equation 8.11 has a real component that goes to $\pm\infty$ as $|t| \to \infty$. Therefore, one of the solutions diverges,

while one approaches 0. Therefore, the IVP cannot be bounded.

Therefore, regardless of $\theta(x)$ chosen (the asymptotic behavior of the curve of integration) the IVP also cannot be bounded and entropy maintaining if $r(t)$ has one or more poles at $t \to \infty$.

□

Lemma 8.1.12. *Given* **IVP** $= \{L, C(x), t_s, \mathbf{IC}, \epsilon\}$, *with* $L(y(t), t) = y''(t) + r(t)y(t) = 0$ *is entropy-maintaining. If* $\lim_{t\to\infty} r(t) = r_\infty \exp(i\phi)$, *where* $r_\infty \in \mathbb{R}$, $\phi \in [0, 2\pi)$, *then* $\lim_{x\to\pm\infty} C(x) = \lim_{R\to\infty} R\exp(i(\pm\pi/2 - \phi/2))$.

Proof. First, by Lemma 8.1.11, $r(t)$ has no poles or zeroes at $t \to \infty$. Therefore, as stated in the Lemma, $\lim_{t\to\infty} r(t) = r_\infty \exp(i\phi)$.

Furthermore, one may construct a series of $r(t)$ as $t \to \infty$:

$$r(t) = r_\infty \exp(i\phi) + \sum_{i=1}^{\infty} C_i t^{-i} \tag{8.13}$$

The WKB approximation holds as $t \to \infty$, because $\frac{dr}{dt} r^{-3/2} \ll 1$.

Therefore, the WKB approximations to two linearly independent solutions of the ODE $y_{\pm,\mathrm{WKB}}$ can be calculated the same way as in Equation 8.11.

As shown in the proof for Lemma 8.1.11, there are two cases for $\lim_{R\to\infty} y_{\pm,\mathrm{WKB}}(R\exp(i\theta))$ when considering Equation 8.12. Note that Equation 8.12 holds when $k = 0$, as is the case for this proof.

If $\pi/2 + \phi/2 + \theta \neq 0, \pm\pi, \ldots$, then the exponent in Equation 8.11 contains a real component that goes to infinity as $|t| \to \infty$. Therefore, the one solution diverges, and the other converges to 0. This is not entropy-maintaining.

Therefore, $\theta = -\phi/2 - \pi/2 + n\pi = -\phi/2 \pm \pi/2 \mod 2\pi$. For $C(x)$ to be proper, one knows that $\lim_{x\to-\infty} C(x) \neq \lim_{x\to\infty} C(x)$. Therefore, $\lim_{x\to\pm\infty} C(x) = \lim_{R\to\infty} \pm R\exp(i(\pm\pi/2 - \phi/2))$.

□

Lemma 8.1.13. *Let* **IVP** $= \{L, C(x), t_s, \mathbf{IC}, \epsilon\}$ *with* $L(y(t), t) = y''(t) + r(t)y(t) = 0$. *Let* $r(t)$ *have* m *finite poles and* m *finite zeroes, and* $\lim_{t\to\infty} r(t) = r_\infty \exp(i\phi)$, *where*

$r_\infty \in \mathbb{R}$, $\phi \in [0, 2\pi)$.

If $C(x)$ *is proper with* $\lim_{x\to\pm\infty} C(x) = \pm \lim_{R\to\infty} R\exp(i(\pm\pi/2 - \phi/2))$, *then* **IVP** *is bounded.*

Proof. Consider x_0 sufficiently large, negative such that $|C(x_0)| \gg |p_i|, |z_i|$ for all i. Consider x_1 sufficiently large, positive such that $|C(x_1)| \gg |p_i|, |z_i|$ for all i. Equation 8.13 approximates $r(t)$ for both $r(C(x_0))$ and $r(C(x_1))$.

Let **IC** be random initial conditions at $t_s = C(x_0)$, and $h(\mathbf{IC}) = l$. First, consider the entropy at $t_1 = C(x_1)$.

Given that L is linear, consider the eigenvalues for the linear system any time t. The largest eigenvalue defines the instantaneous rate of divergence of the ODE (i.e., the Lyapunov exponent). Therefore, if one can identify the maximal instantaneous eigenvalue of L on the interval $C([x_0, x_1])$, then this bounds the overall rate of divergence of the system. One can find a maximal eigenvalue on the interval $C([x_0, x_1])$ and use this to bound $|y(t_1)|, |y'(t_1)|$ given bounds on $|y(t_0)|, |y'(t_0)|$ as follows.

Consider transforming the ODE $L = y''(t) + r(t)y(t)$ to be in terms of the parameter x (note that in Equation 8.14, $y'(C(x)) = \frac{d}{dx}y(C(x))$.):

$$y''(C(x))(C'(x))^2 + y'(C(x))C''(x) = r(C(x))y(C(x)) \tag{8.14}$$

The eigenvalues of this ODE (in terms of x) are:

$$\lambda_\pm = \frac{C''(x)}{2(C'(x))^2} \pm \frac{1}{2}\sqrt{\frac{(C''(x)^2}{(C'(x))^4} + 4\frac{r(C(x))}{(C'(x))^2}} \tag{8.15}$$

Because $C(x)$ is proper, $C'(x)$ is never zero, nor does it asymptotically approach zero (since it is analytic, and $\lim_{x\to\pm\infty} |C(x)| = \infty$). Therefore, there exists some $\delta > 0$: $\forall x,\ |C'(x)| > \delta$. Moreover, since $C(x)$ is analytic for all $x \in \mathbb{R}$, $|C(x)'|$, $|C''(x)|$ are bounded, finite on the finite interval $[x_0, x_1]$.

Finally, because $C(x)$ is proper, it does not contain any of the poles of $r(t)$. Therefore, $|r(C(x))|$ is also bounded on the interval $[x_0, x_1]$.

One may conclude, therefore, that $|\lambda_\pm|$ from Equation 8.15 are also both bounded

by some finite λ_{MAX}.

The maximum value of $|y(t_1)|, |y'(t)|$ is therefore bounded by $|y(t_0)| \exp(|x_0 - x_1|\lambda_{\text{MAX}})$, and $|y'(t_0)| \exp(|x_0 - x_1|\lambda_{\text{MAX}})$ respectively.

Therefore, $|y(t_0)|, |y'(t_0)|, |y(t_1)|, |y'(t_1)|$ are all bounded, finite.

Now, consider $|y(C(x))|, |y'(C(x))|$ as $|x| \to \infty$. Recognize that for sufficiently large t, the WKB conditions hold (as observed in Lemma 8.1.12, $\frac{dr}{dt} r^{-3/2} \ll 1$). Therefore, the WKB approximations to two linearly independent solutions of the ODE $y_{\pm,\text{WKB}}$ can be calculated the same way as in Equation 8.11.

For sufficiently negative x_0 and positive x_1, the behavior of $C(x)$ at x_0 and x_1 is represented by $C(x) = R(x) \exp(i\theta(x))$, with $\theta(x) = \sum_{i=0}^{\infty} \theta_i x^{-i}$. Plugging this into Equation 8.12 with $k = 0$:

$$y_{\pm,\text{WKB}}(C(x)) \approx \left(r_\infty \exp(i\phi) + O(R(x)^{-1})\right)^{-1/4} \times \exp\left(\pm i(\sqrt{r_\infty} \exp(i\phi/2) \times R(x) \exp(i\theta(x))) + O(R(x)^{1/2})\right) \tag{8.16}$$

First, recognize that the multiplicative prefactor has bounded behavior as $x \to \pm\infty$, as it only depends on $O(R(x)^{-1})$, and $\lim_{x\to\infty} O(R(x)^{-1}) = 0$ because $C(x)$ is proper.

Second, recognize $\lim_{x\to\pm\infty} \theta(x) = \pm\pi/2 - \phi/2$ by the statement of the Lemma. Therefore, write $\theta(x) = \pm\pi/2 - \phi/2 + O(x^{-1})$. Now, plug this into the term $\sqrt{r_\infty} \exp(i\phi/2) \times R(x) \exp(i\theta(x)))$ to obtain:

$$\sqrt{r_\infty} \exp(i\phi/2) \times R(x) \exp(i(\pm\pi/2 - \phi/2 + O(x^{-1}))) = \pm iR(x)\sqrt{r_\infty} \exp(iO(x^{-1}))$$

Plug this into Equation 8.16 to obtain the following approximation for $y(C(x))$ in the limit of large positive or negative x:

$$y_{\pm,\text{WKB}}(C(x)) \approx \left(r_\infty \exp(i\phi) + O(R(x)^{-1})\right)^{-1/4} \times \exp\left(\pm iR(x)\sqrt{r_\infty} \exp(iO(x^{-1}))\right) \tag{8.17}$$

Note that the exponent term approaches a limit cycle, as its argument is approaching purely imaginary. Since $y_{\pm,\text{WKB}}(C(x_0))$ and $y_{\pm,\text{WKB}}(C(x_1))$ are both bounded,

$y_{\pm,\text{WKB}}(C(x))$ remains bounded, entropy maintaining for $x \to \pm\infty$.

□

The necessary and sufficient conditions are now formalized into the following theorem.

Theorem 8.1.14. *The following conditions are necessary and sufficient for an IVP to be entropy-maintaining and bounded.*

- $\lim_{x\to\pm\infty} y(C(x))$ *does not exist (DNE).* $y(C(x))$ *is bounded and approaches a limit cycle.*
- $r(t)$ *has a finite, equal number of poles and zeroes.*
- *If* $\lim_{t\to\infty} r(t) = r_\infty \exp(i\phi)$, *for* $r_\infty \in \mathbb{R}$, *then* $\lim_{x\to\pm\infty} C(x) = \pm \lim_{R\to\infty} R\exp(i(\phi/2 - \pi/2))$. *This is depicted in Figure 8-1.*

Proof. Lemmas 8.1.10, 8.1.11, and 8.1.12 show that the above conditions are necessary for an IVP to be entropy-maintaining given Definition 8.1.4.

Lemma 8.1.13, shows that the above conditions are sufficient for an IVP to be entropy maintaining given Definition 8.1.4.

□

A "good" IVP and associated curve of integration are shown pictorially in Figure 8-1.

8.1.3 Equivalence of IVPs

Now that a notion of entropy-maintaining, bounded IVPs is established, there is an equivalency class among these IVPs. This is provided in Definition 8.1.15. Note that this incorporates the notions of translation, rotation, and scaling of the solution $y(t)$.

Definition 8.1.15. *Let* $[L]$ *be the equivalence class of* L. *Any* $L' \in [L]$ *has the property that if* $y_1(t), y_2(t)$ *are two linearly independent solutions of* L, *then* $y_1(at + b)$, $y_2(at + b)$ *are two linearly independent solutions of* L' *for some* $a, b \in \mathbb{C}$.

Lemma 8.1.16. *Given* **IVP** *with ODE* $L = y''(t) + r(t)y(t)$*, there is an equivalent* $\mathbf{IVP_T}$ *with ODE* $L = y_T''(t) = r_T(t)y_T(t)$ *under Definition 8.1.15 such that* $r_T(t) = a^{-2}r(at+b)$*, for any* $a, b \in \mathbb{C}$.

Proof. Consider **IVP**, with $L = y''(t) + r(t)y(t)$. By Definition 8.1.15, there exists an equivalent $\mathbf{IVP_T}$ such that $y(t)$ is transformed to $y(at+b)$.

Rewriting the ODE:

$$a^2 y(at+b)'' = r(at+b)y(at+b)$$

Assigning $y_T(t) = y(at+b)$ yields $y_T''(t) = r_T(t)y_T(t)$, where $r_T(t) = a^{-2}r(at+b)$

□

Corollary 8.1.17. *Any bounded, entropy-maintaining* **IVP** *with* $L = y''(t) + r(t)y(t)$ *is equivalent to some* $\mathbf{IVP_N}$ *with* $L_N = y_N''(t) + r_N(t)y_N(t)$*, where* $\lim_{t\to\infty} r_N(t) = 1$.

Proof. If $\lim_{t\to\infty} r(t) = r_\infty$, then by Lemma 8.1.16, one may transform **IVP**. Setting $a = 1/r_\infty$ results in $\mathbf{IVP_N}$ with $\lim_{t\to\infty} r_N(t) = 1$.

□

The above equivalence class allows the rotation and translation of entropy-maintaining, bounded ODEs. This will be important, because it allows one to "stitch together" two separate IVPs into a single IVP in a coherent way that maintains the bounded, entropy maintaining properties of both ODEs. This is formalized in Lemma 8.1.18, and shown pictorially in Figure 8-2.

Lemma 8.1.18. *Given n IVPs* $\mathbf{IVP}_j$ *(with j from 1 to n) with m poles, m zeroes that are each entropy-maintaining, let* $R \in \mathbb{R} : \forall\ z_{i,j},\ |z_{i,j}| \leq R,\ \forall\ p_{i,j},\ |p_{i,j}| \leq R$*, where* $z_{i,j}$*,* $p_{i,j}$ *are the sets of zeroes and poles in* $\mathbf{IVP}_j$ *respectively. Let* $[z_{i,j}], [p_{i,j}]$ *be the sets of zeroes and poles in IVPs that are equivalent to* $\mathbf{IVP}_j$ *by Definition 8.1.15.*

There exists a third entropy-maintaining IVP $\mathbf{IVP_{TOT}}$ *with* $n \times m$ *zeroes at* $z_{i,\text{tot}}$ *and* $n \times m$ *poles at* $p_{i,\text{tot}}$ *that has the following property: Define* $z_{i,t}$ *to be the m closest zeroes to* $C(t)$*. Define* $p_{i,t}$ *similarly.*

There exists $t_j \in \mathbb{R} :\ z_{i,t_j} = [z_{i,j}],\ p_{i,t_j} = [z_{i,j}]$.

Proof. Choose some finite $T \gg R$.

Use Corollary 8-2 $\mathbf{IVP}_j$ to $\mathbf{IVP}_{j,\mathrm{N}}$. Therefore, the associated ODE of each $\mathbf{IVP}_{j,\mathrm{N}}$ has the property that $\lim_{x\to\infty} r_{j,\mathrm{N}}(C(x)) = 1$. Second, use Lemma 8.1.16 to translate each $\mathbf{IVP}_{j,\mathrm{N}}$ by $b_j = j \times T$. Denote this set of IVPs as $\mathbf{IVP}_{j,\mathrm{NT}}$.

These IVPs are still independent, but they can be merged into a single IVP by collecting all of the poles/zeroes into a single IVP and connecting the curves of integration into a single curve.

If the final IVP is denoted as $\mathbf{IVP_{TOT}}$ then since each IVP is normalized:

$$r_{\mathrm{TOT}}(t) = \prod_{j=1}^{n} r_{j,\mathrm{NT}}(t)$$

Connecting the curves is also simple. The curve of each ODE is proper. Therefore, one may truncate the curves in between IVPs at radius R from its translated origin and connect the curves in a smooth way such that Definition 8.1.5 holds. Note that because $y(t)$ is analytic everywhere except at the poles of r_{TOT} and ∞, one may deform this curve with significant freedom. Simply ensure that the connecting curve does not encircle any poles of r_{TOT}.

Therefore, this constructs a single IVP that is by definition bounded and entropy-maintaining. Also, by construction, there exist points (at the translated origin of each IVP) where $t_j \in \mathbb{R} :\ z_{i,t_j} = [z_{i,j}],\ p_{i,t_j} = [z_{i,j}]$.

□

8.2 Sampling Initial Value Problems

Now that the relevant properties of IVPs have been established, the problem of solving IVPs must be formally stated in the language of complexity theory.

Definition 8.2.1 (Initial Value Problems as a Decision Problem). *Given constants* ϵ, $C(x)$, t_s *that each have a finite representation, consider a family*[2] *of* $\mathbf{IVP}_{\epsilon,C(x),t_s}$,

[2]The choice of this family is the subject of this section.

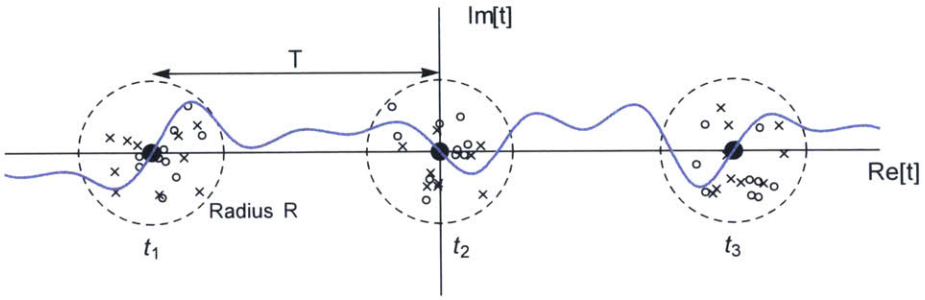


Figure 8-2: Example of three connected IVPs. The dotted circles denote radius R around the origin of the IVP. These origins are translated to points t_1, t_2, t_3 that are distance $T \gg R$ apart. This locally confines the effect of each IVP. Corollary allows one to rotate/translate these ODEs such that the curves of integration can be stitched together in the region greater than R away from any of t_1, t_2, t_3.

with these parameters fixed. Next, consider a pair $\mathbf{Rslt} = (t_f, y(t_f))$, *with* $t_f \in C(x)$, *and* $y(t_f) \in \mathbb{C}$.

Let the exact solution of **IVP** *at* $t = t_f$ *be* $Y(t_f)$. *Consider a language* $\mathcal{L}$ *such that* $(\mathbf{IVP}, \mathbf{Rslt}) \in \mathcal{L}$ *if and only if* $|Y(t_f) - y(t_f)| < \epsilon$. *Let the equivalent decision algorithm* $f(\mathbf{IVP}, \mathbf{Rslt})$ *accept if and only if* $(\mathbf{IVP}, \mathbf{Rslt}) \in \mathcal{L}$.

There are several clarifications that should be made at this point. First, the language $\mathcal{L}$ has words $(\mathbf{IVP}, \mathbf{Rslt})$. These words are not discrete, even though $\epsilon, C(x), t_s$ are fixed parameters of the language. As a result, since no real (or complex) number can be represented as a string, there is no Turing machine that could hope to compute on this input. Furthermore, the ODE operator $\mathcal{L}$ must be assigned a general representation.

First, let the ODE operator L be of the form $y''(t) = r(t)y(t)$, with $r(t)$ defined as follows (note that r_{lim} is usually set to 1 via Corollary 8-2):

$$r(t) = r_{\text{lim}} \times \frac{\prod_{i=1}^{m_z} 1 - \frac{t-t_s}{z_i - t_s}}{\prod_{i=1}^{m_p} 1 - \frac{t-t_s}{p_i - t_s}} \tag{8.18}$$

The representation in Equation 8.18 is important, as it allows L to be represented as a vector of complex values: $\mathbf{p}$, $\mathbf{z}$, r_{lim}, where $\mathbf{p}$, $\mathbf{z}$ are sets of the poles and zeroes of

Equation 8.18, respectively.

Next, one must identify a representation of complex numbers that is compatible with finite computation. Two different approaches will be taken to address this issue depending on whether the computer in question is analog or digital.

I will start by taking an approach similar to Kawamura [83]. A complex value is represented to the Turing machine as an oracle that, when queried with a positive integer n, returns a string representation of the associated analog value that is accurate to within[3] $\delta = 2^{-n}$. For a digital computer, this implies that a complex number may simply be described as a finite bitstring that describes the number to the correct precision. For an analog computer, treat the above oracle as a continuous random variable with differential entropy $-n$.

The following analysis is purely from the perspective of a digital computer, therefore, treat complex numbers as finite bitstrings with some encoding. However, later in this work (cf. Definitions 8.5.1 and 8.5.2) both analog and digital aspects of this representation will be studied.

Now, restrict the domain of the possible complex values to be considered for use as parameters in $r(t)$ and as elements of **IC**. First, recognize that both **IC** and parameters in $r(t)$ must be finite. Moreover, since $r(t) \in \mathbb{C}(t)$, there are a finite number of poles/zeroes in $r(t)$.

Therefore, any set of finite complex values are allowed to define $r(t)$ and **IC**. In order to pick these parameters randomly, consider that they are distributed uniformly on a disc of radius R. For sufficiently large R, I posit that this distribution contains all possible parameter sets of interest for a given number of poles/zeroes. Moreover, the uniform distribution is chosen on this disc because it maximizes the entropy of the distribution.

Definition 8.2.2 (Family of Initial Value Decision Problems). *Let* $U_{R,m,\delta} = \{\mathbf{IVP}_{\epsilon,C(x),t_s}, \mathbf{Rslt}_{t_f}\}$ *be a family of initial value problems as in Definition 8.2.1, where* $r(t)$ *contains*

[3]This approach is deeply related to the notion of differential entropy (Definition 8.1.1). The notion of accuracy to within 2^{-n} is equivalent to having a probability distribution over $\mathbb{C}$ in a disc of radius 2^{-n} around a point. The information required to encode the complex value is therefore this small probability distribution minus the differential entropy of the distribution of all possible complex values.

m poles, m zeroes, distributed on a disc of radius R in the complex plane centered around $t = 0$. All complex values have accuracy δ. The IVP requires accuracy ϵ, and integrates over curve $C(x)$ between t_s, and t_f.

This family of IVPs can be randomly sampled by a finite computer in polynomial in $m, \log(\delta), \log(R)$.

8.3 Formalization of Numerical Integration

Now that there is a clear notion of initial value problems, I posit a formalization of numerical integration algorithms. In essence a numerical integration algorithm is decomposed into the composition of two algorithms: the expansion algorithm Exp, and the integration algorithm NI. The integration algorithm calls Exp as a subroutine.

The Exp algorithm has only local knowledge of the ODE, while the overall numerical integration algorithm may optimize globally. This reflects a broad class of numerical integration algorithms that contain single and multi-step methods, different expansion routines (e.g., Taylor expansion, Padé expansion, etc.), fixed/variable step size methods, etc.

Since the Exp only has local knowledge of the system's behavior, restrict its knowledge of the ODE (described by $r(t) \in \mathbb{C}(t)$ as in Definition 8.1.4). This is done by providing Exp with only access to the first $k - 2$ derivatives of $r(t)$ at t_s.[4] This restriction reflects the current approach taken by all numerical expansions used for numerical integration [34].

Exp is not provided with a full description of the global behavior of $r(t)$. Restricting the knowledge of Exp provides a clean separation between Exp and NI. In particular, if Exp has global knowledge of $r(t)$, then it could conceivably implement its own numerical integration algorithm as a subroutine and achieve an unbounded region of convergence R for any expansion order k.

[4]The ODE is second order, so only $k - 2$ derivatives of $r(t)$ are needed to compute the first k derivatives of $y(t)$ at t_s.

Definition 8.3.1. *A single-step expansion algorithm family*

$$\mathsf{Exp}_{k,\epsilon}(\{r'(t_s),\ldots,r^{(k-2)}(t_s)\},t_s,\mathbf{IC}) \rightarrow (f(t),\mathbf{R})$$

is indexed by the expansion order k and the maximum tolerable error ϵ. It takes the first $k-2$ derivatives of $r(t_s)$, the initial conditions $t_s, \mathbf{IC}$. It returns a function $f(t)$ with the guarantee:

$$\left|\frac{y(t)-f(t)}{y(t)}\right| < \epsilon \, \forall t \in \mathbf{R} \tag{8.19}$$

Finally, let $\mathsf{Exp}_{k,\epsilon}$ run in $O(\mathsf{poly}(k,-\log(\epsilon))$.

The above definition may be bolstered to include multi-step methods:

Definition 8.3.2. *A multi-step expansion algorithm family*

$$\mathsf{Exp}_{k,\epsilon}(\{r'(t_s),\ldots,r^{(k-2)}(t_s)\},t_s,\mathbf{IC}_0,\{r'(t_1),\ldots,r^{(k-2)}(t_1)\},t_1,\mathbf{IC}_1,\ldots) \rightarrow \{f(t),\mathbf{R}\}$$

is identical to a single-step expansion algorithm (Definition 8.3.1 save that it has knowledge of $r(t)$, $y(t)$ at some number of additional points $t_i \in \mathbb{C}$).

These expansion algorithms are used in sequence to construct a numerical integrator, defined below:

Definition 8.3.3. *A numerical integrator $\mathsf{NI}_{\mathsf{Exp},\mathbf{k}}(\mathbf{IVP},t_f)$ is a family of algorithms that integrates* **IVP** *from t_s to t_f along $C(x)$.*

The integrator uses expansion algorithm family Exp*, and has a list of integers* $\mathbf{k}$*.* NI *then follows the procedure in Algorithm 3.*

The integrator uses function $x \leftarrow \mathsf{Step}(C(x),\mathbf{R})$ to determine the next point of expansion. Typically, this algorithm chooses maximal x such that $C(x) \in \mathbf{R}$.

The complexity of computing a numerical integration may be considered as the sum of costs of each expansion, if the costs of the integrator is small. Therefore, define the "optimal" numerical integrator with respect to the optimization of this cost.

Algorithm 3 Form of a Numerical Integration Algorithm

```
1: procedure NI_{Exp,k}(IVP, t_f)
2:     Set i = 0
3:     Set t_c = t_s
4:     Set Y = IC
5:     while t_i ≠ t_f do
6:         Call (f(t), R) = Exp_{k_i,ε}({r'(t_i), ..., r^{(k-2)}(t_i)}, t_i, Y)
7:         i = i + 1
8:         if t_f ∈ R then
9:             t_i = t_f
10:        else
11:            Compute x_new = Step(C(x), R)
12:            t_i = C(x_max)
13:        end if
14:        Y = (f(t_i), f'(t_i))
15:    end while
16:    return Y
17: end procedure
```

Definition 8.3.4. *Define the "optimal numerical integration algorithm"* $\mathsf{NI}^{\mathsf{OPT}}$ *for* **IVP** *as a numerical integration algorithm according to Definition 8.3.3 with the following properties.*

Let the computational cost of the expansion be $\mathrm{Cost}(\mathsf{Exp}_{k,\epsilon})$. *Define a family of numerical integrators indexed by the multiset of integers* $\mathbf{k}$. *Let the cost of the overall numerical integration algorithm be* $\mathrm{Cost}(\mathsf{NI}_{\mathsf{Exp},\mathbf{k}}) = \sum_{k \in \mathbf{k}} \mathrm{Cost}(\mathsf{Exp}_{k,\epsilon})$.

First, require that the pair $\mathbf{Rslt} = (t_f, \mathsf{NI}_{\mathsf{Exp},\mathbf{k}}(\mathbf{IVP}, t_f))$ *have the property that* $(\mathbf{IVP}, \mathbf{Rslt}) \in \mathcal{L}$, *where* $\mathcal{L}$ *defined in Definition 8.2.1.*

The optimal numerical integration algorithm is an algorithm with the property:

$$\mathrm{Cost}(\mathsf{NI}^{\mathsf{OPT}}) = \mathrm{Inf}_{\mathbf{k}} \mathrm{Cost}(\mathsf{NI}_{\mathsf{Exp},\mathbf{k}}(\mathbf{IVP}, t_f))$$

8.4 Conjecture

I now formalize a conjecture regarding the difficulty of simulating differential equations.

Conjecture 8.4.1 (Primary Difficulty Conjecture). *Given* $U_{R,m,\delta} = \{\mathbf{IVP}_{\epsilon,C(x),t_s}, \mathbf{Rslt}_{t_f}\}$

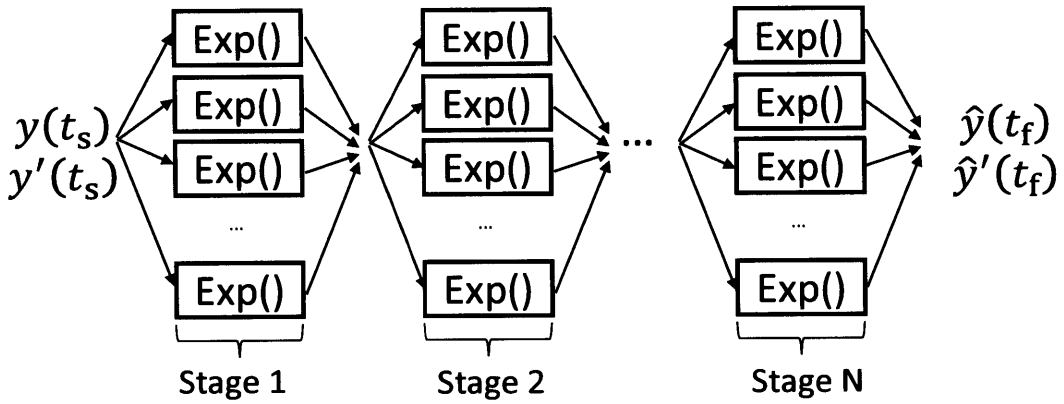


Figure 8-3: Form of a numerical integrator in Conjecture 8.4.1. Note that each individual stage may call Exp many times in parallel with varying expansion orders for various values of t.

as in Definition 8.2.2 with $t_f = 0$ (note that $t_f \in C(x)$), any algorithm that correctly solves a subset of $U_{R,m,\delta}$ whose volume is non-negligible in R^{2m} (the total probability volume of the family $U_{R,m,\delta}$) will be a numerical integrator conforming to Definition 8.3.3.

Note that Conjecture 8.4.1 restricts the possible form of any numerical integration algorithm that can be used to solve the IVP to be of the form in Figure 8-3. The critical restriction is on the expansion algorithm used by the numerical integrator in Definition 8.3.3 – that it only has knowledge of the first k derivatives of $r(t)$ at the point of expansion. I now discuss the implications of this restriction.

8.4.1 Conjecture Discussion

From a practical perspective, all known ODE integrators that the author is aware of can be described as an element or subclass of Definition 8.3.3. This provides empirical evidence that the conjecture is reasonable. However, it is still instructive to fully understand the implications of this restriction.

Consider the restriction on the form of expansion algorithm via Definition 8.3.1 – that Exp knows only the first $k-2$ derivatives of $r(t)$. This restriction is chosen to enforce that an order-k expansion can only compute an approximation that matches

the exact solution to order k.

Given the ODE $y''(t) = r(t)y(t)$, the expansion algorithm Exp cannot compute $y^{(k+1)}(t)$ without knowledge of $r^{(k-1)}(t)$. One therefore may consider higher-order coefficients of $r(t)$ as random variables, and Exp may attempt to estimate these random variables based on its knowledge of the structure of the space of ODEs. Informally speaking, the better an expansion algorithm is at estimating these higher orders, the better expansion it returns.

At first, the restriction on knowledge of $r(t)$ may seem onerous. However, recall that a Padé expansion of a function converges in measure to any meromorphic function for sufficiently large k. Therefore, sufficient local information can be used to derive global behavior of the function.

However, I will analyze the implications of the conjecture at some finite expansion order k.[5]

For finite k, Conjecture 8.4.1 implies that there will always be uncertainty in the approximation provided by the expansion algorithm. This uncertainty is bounded using the same notion of differential entropy discussed earlier.

Empirical Observation 8.4.2. *Given $U_{R,m,\delta} = \{\mathbf{IVP}_{\epsilon,C(x),t_s}, \mathbf{Rslt}_{t_f}\}$, as in Definition 8.2.2, let $t_s = 0$, and $t_f > R$, with $t_s, t_f \in C(x)$.*

There exists a choice of R, m, t_f such that $h(y(t_f))/(-\log(\epsilon))$ is greater than 1 bit with probability non-negligible in m given knowledge of the first $2m-1$ derivatives of $r(t)$ at $t = t_s$.

Empirical Observation 8.4.2 formalizes the intuitive statement that a randomized initial value problem retains a non-negligible amount of entropy in its solution even when most (in this case all but one) of the derivatives of its coefficient ($r(t)$) are given. The expression $h(y(t_f))/(-\log(\epsilon))$ is the entropy for a discrete random variable generated by measuring a continuous random variable $y(t_f)$ to precision ϵ.

Therefore, if an analog system instantiating a random IVP from $U_{R,m,\delta}$ may be reliably measured to precision ϵ, a digital simulation that knows only the first $2m-$

[5]Later, I will use k as "security parameter" wherein any adversary that hopes to simulate the system faster must be able to *quickly* compute expansions of order greater than k.

1 derivatives of $r(t)$ cannot compute the least significant bit of the measurement, because the least significant bit of this analog measurement is random (has non-negligible entropy) to the digital simulation.

First, note that if $2m$ derivatives of $r(t)$ are given to arbitrary precision, then these can be used (via Padé expansion or similar) to identify the location of all of the poles/zeroes of $r(t)$. Once an expansion algorithm has global knowledge of $r(t)$, then $y(t_f)$ has no entropy remaining (this is why Conjecture 8.4.1 is made).

This observation is empirical, because $\mathsf{Pr}(y(t_f))$ given knowledge of some of the derivatives of $r(t_s)$ is exceedingly complex. This is discussed in the next subsection.

8.4.2 Statistical Justification

The problem addressed in this section is how to compute $\mathsf{Pr}(y(t_f))$ given the first k derivatives of $r(t)$ at $t = t_s$, where $y''(t) = r(t)y(t)$ with the guarantee that $r(t) \in \mathbb{C}(t)$ with m poles and m zeroes.

Before beginning the analysis, recognize that the IVP is defined in terms of the locations of the poles/zeroes of $r(t)$ (e.g., $U_{R,m,\delta}$ from Definition 8.2.2), while the performance of an expansion is determined by the expansion order, or equivalently the number of derivatives of $r(t)$ known by the expansion algorithm.

The accuracy of any algorithm's estimate of $y(t_f)$ is determined by how accurately such an algorithm can estimate the locations of the poles/zeroes of $r(t)$. Therefore, one may equivalently compute $\mathsf{Pr}(r(t))$ given the same information as above. However, recognize that $r(t)$ drawn from $U_{R,m,\delta}$ has the restriction that all poles/zeroes must be inside a disc of radius R around $t = 0$. This restriction must be taken into account, so in essence, one must compute the probability distribution of the locations of the poles/zeroes of $r(t)$ given knowledge of the first k derivatives of $r(t)$.

This problem has been extensively studied in the context of numerical approximation theory and generally the result is that pole/zero locations can be *very sensitive* to small changes in the higher order derivatives, and their behavior is very complex [171, 76]. However, because I am performing a statistical analysis, to show Empirical Observation 8.4.2, I do not need to study the problem analytically. I instead randomly

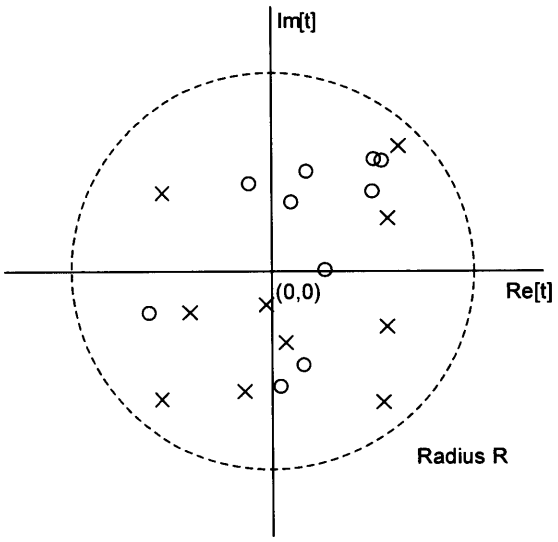


Figure 8-4: Example randomly selected pole/zero configuration for $R = 10$, $m = 10$, $\delta = 10^{-5}$.

sample members of $U_{R,m,\delta}$, and numerically compute the probability distribution and its entropy. However, one must be careful in the analysis to account for numerical errors.

I use a Monte Carlo method, beginning by sampling a random IVP from $U_{R,m,\delta}$. For this study, I choose $R = 10$, $m = 10$, $\delta = 10^{-5}$. The chosen $r(t)$ has poles at $\mathbf{p_{orig}}$ and zeroes at $\mathbf{z_{orig}}$. For example, configure the pole/zero configuration in Figure 8-4.

I then compute the order $k = 2m - 1$ series expansion of $r(t)$ at $t = t_s$, denoted as $\text{rser}(t) = \sum i = 0^{2m-1} C_i t^i$. Recall that $k = 2m - 1$ because this is the maximal k for which there is still entropy in $r(t)$. This is because an algorithm given $k = 2m$ derivatives of $r(t)$ to arbitrary, finite precision can compute (using Padé expansion) the locations of the poles/zeroes of $r(t)$. Note that setting $k = 2m - 1$ is a *worst case* choice of k. In most cases, a numerical integration algorithm will use an expansion order $k \ll 2m-1$. However, the below analysis becomes exponentially difficult in l for studying a maximum expansion order of $k = 2m - l$. Moreover, from a cryptographic perspective, the worst case is most interesting.

The next step, intuitively speaking, is to construct the set of possible pole/zero

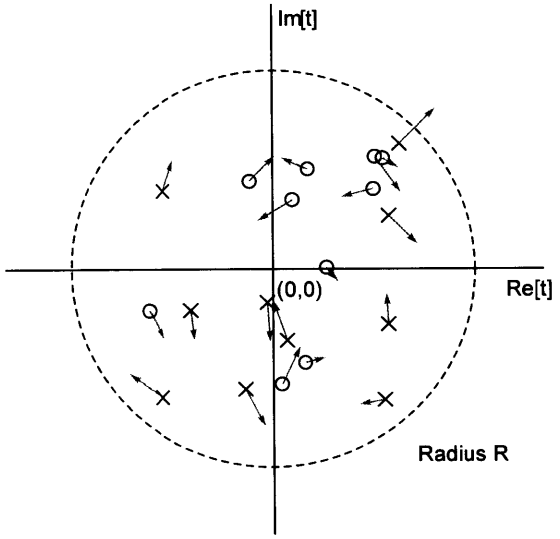


Figure 8-5: Example dependence of the pole/zero configuration on a small change in C_{2m}. Although the pole/zero locations can be very sensitive to changes in C_{2m} (i.e., see [171]), there is still a smooth dependence. Note, however, that as C_{2m} changes, it is possible for one or more poles/zeroes to leave the disc of radius R. This would be result in the configuration not being a valid sample from $U_{R,m,\delta}$. This must be addressed.

configurations that satisfy the requirement that the first k derivatives match the given derivatives (the requirement that these pole-zero configurations correspond to $r(t)$ that are elements of $U_{R,m,\delta}$ is imposed later). In order to do this, recognize that the first $2m-1$ derivatives of $r(t)$ are given, and therefore, the set of possible functions $r(t)$ may be given by:

$$\{\mathsf{Pade}_{m,m}\left(\mathrm{rser}(t) + C_{2m}t^{2m}\right)\} : C_{2m} \in \mathbb{C} \tag{8.20}$$

Further, recognize that the pole-zero configuration smoothly depends on C_{2m}, and that any region of C_{2m} corresponds to a set of pole-zero configurations. This dependence is shown in Figure 8-5.

Observe that the dependence of the pole/zero configuration on C_{2m} may be viewed as a one-dimensional curve over a $2m$ dimensional vector-space. In particular, observe that each pole/zero location is an element of a $2m$-dimensional vector, which is an

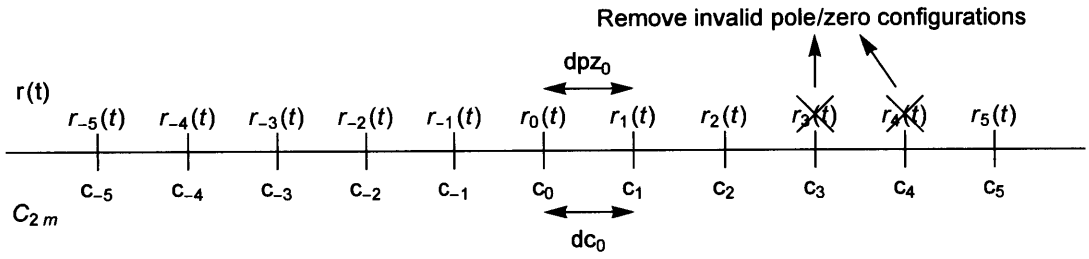


Figure 8-6: A depiction of the dependence of the pole/zero configuration on C_{2m}. As C_{2m} changes by dc_i, the pole/zero configuration changes by dpz_i (computed with Equation 8.21). Note that certain configurations may be invalid because one or more poles/zeroes fall outside the disc of radius R. This is shown with (for example) $r_3(t)$ and $r_4(t)$.

overall function of a single variable C_{2m}, and this represents a one-dimensional path in a $2m$ dimensional vector space.

As a result, an interval $C_{2m} \in [c_0, c_1)$ with $\mathrm{dc}_0 = c_1 - c_0$ has a corresponding interval of pole-zero configurations, each representing a possible choice for $r(t)$. This is depicted in Figure 8-6. If the interval of pole-zero configurations is such that for all $C_{2m} \in [c_0, c_1)$, all poles/zeroes are inside the disc of radius R, then all configurations are allowed and equally likely. The probability therefore is related to the "distance" of the interval of pole-zero configurations (denoted dpz_0 in Figure 8-6).

Since the pole-zero configuration is a one-dimensional path in a $2m$-dimensional vector space, the distance metric is the Euclidean distance in a $2m$-dimensional space, computed as follows where **pz1**, **pz2** are sets of poles/zeroes.

$$\mathrm{Dist}(\mathbf{pz1}, \mathbf{pz2}) = \sqrt{\sum_i |\mathbf{pz1}_i - \mathbf{pz2}_i|^2} \tag{8.21}$$

Note that not all intervals have the property that all of the poles/zeroes are located on the disc of radius R around $t = 0$ during the interval. In this case, the interval is illegal, as the corresponding $r(t)$ is not in $U_{R,m,\delta}$, which is required by the statement of the problem. Therefore, drop the interval from further computations.

Also note that c_i as shown in Figure 8-6 is defined for all $i \in \mathbb{Z}$. This cannot be computed statistically. Fortunately, it is not necessary, as when $|i|$ becomes large,

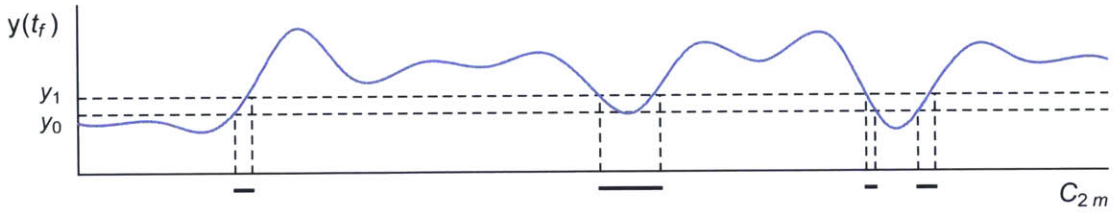


Figure 8-7: Depiction of the dependence of $\mathsf{Pr}(y(t_f))$ given $\mathsf{Pr}(C_{2m})$. In particular, the demonstration of the computation of $\mathsf{Pr}(y(t_f) \in [y_0, y_1)$. The transformation is given in Equation 8.23.

$r(t)$ converges into a stable pole/zero configuration. This intuitively makes sense, because if C_{2m} is much greater than all other series coefficients (which is true for large $|i|$), this corresponds to a single pole/zero configuration. As a result, dpz_i goes to 0 as i goes to $\pm\infty$. As such, simply truncate the computation at some large $|i|$, as pole/zero configurations corresponding to intervals with larger $|i|$ are very unlikely (more on this below).

One can now compute $\mathsf{Pr}(C_{2m} \in [c_i, c_{i+1}])$ given the first $2m-1$ derivatives of $r(t)$. Recognize that the probability of observing C_{2m} in a given interval $[c_0, c_1)$ is given by Equation 8.22. The sum is taken over all valid intervals.

$$\mathsf{Pr}\left(C_{2m} \in [c_0, c_1)\right) \approx \frac{\mathrm{dpz}_0}{\mathrm{dc}_0} \Bigg/ \sum_i \frac{\mathrm{dpz}_i}{\mathrm{dc}_i} \tag{8.22}$$

This discrete probability distribution now approximates the PDF for continuous random variable $\mathsf{Pr}(C_{2m})$. The next step is to compute the corresponding value of $y(t)$ at the point of evaluation t_f for each value of C_{2m}. I use this to compute an approximation $\mathsf{Pr}(y(t_f))$ given the first $2m-1$ derivatives of $r(t)$.

The method of this computation is depicted in Figure 8-7, and given in Equation 8.23.

$$\mathsf{Pr}(y(t_f) \in [y_0, y_1)) = \mathsf{Pr}\left(C_{2m} \in \{c : y_0 \leq y(t_f) < y_1 | C_{2m} = c\}\right) \tag{8.23}$$

This finally gives the probability distribution for $y(t_f)$ given the first $2m-1$ derivatives of $r(t)$. Finally, compute the entropy of this distribution. The differential

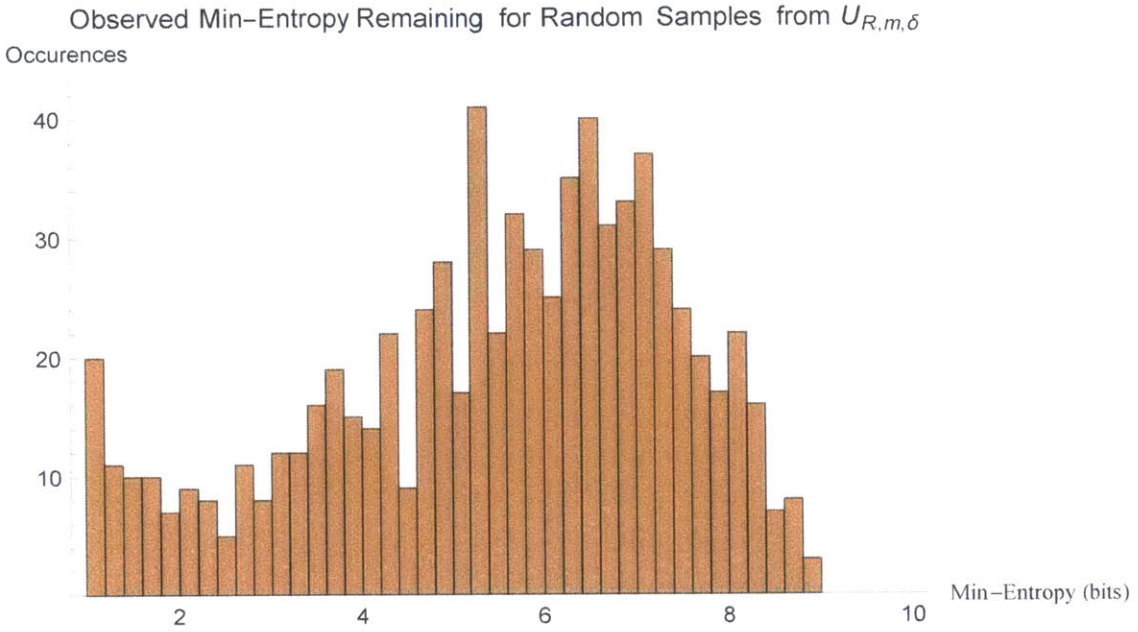


Figure 8-8: A histogram generated by measuring the min-entropy of $y(t_f)$ given the knowledge of the first $2m-1$ derivatives of $r(t_s)$ for 1000 random samples from $U_{R,m,\delta}$. These data are collected by assuming a measurement precision of 10 bits (and therefore a maximum entropy of 10 bits). Out of 1000 samples, each system had significant min-entropy. Therefore, there is evidence that Empirical Observation 8.4.2 is true.

entropy is too optimistic given the wide range of magnitude possibilites for $y(t_f)$; it assumes a constant measurement precision regardless of the amplitude of $y(t_f)$. In general high-speed analog systems can only measure a few bits of precision. For the purposes of this experiment, define ϵ relative to $|y(t_f)|$. I choose 10 bits of precision, implying that for a given sample $\epsilon = |y(t_f)|/1024$.

As such, consider the entropy of $y(t_f)$ relative to a measurement precision of $y(t_f)/2^{10}$. This is done by putting the distribution of $\mathsf{Pr}(y(t_f))$ into 2^{10} buckets and calculating the entropy of that set.

The above algorithm was run on 1000 randomly sampled elements of $U_{R,m,\delta}$ with $R = 10, m = 10, \delta = 10^{-5}$. Based on the above analysis, I computed the distribution in Figure 8-8 of remaining entropy after receiving the first $2m-1$ derivatives.

It was found that greater than 1 bit of entropy remained with non-negligible probability. In fact, out of maximum possible entropy of 10 bits, a substantial fraction

of entropy typically remained, even though $2m-1$ derivatives of $r(t_s)$ were given. Therefore, one concludes that $y(t_f)$ has several bits of entropy after the first $2m-1$ derivatives of $r(t_s)$ are revealed (Empirical Observation 8.4.2 is true).

8.4.3 Error Analysis

There are several possible sources of error in the calculation of the entropy of $y(t_f)$ given the first $2m-1$ derivatives of $r(t)$. They fall into the following buckets:

- Roundoff error, numeric error, and arithmetic precision.
- Numeric integration error in Equation 8.22.
- Numeric integration error in Equation 8.23.

Roundoff Error: In the above algorithm, I determine (a) valid pole-zero configurations and (b) interval distance in the $2m$ dimensional space of pole/zero configuration by:

1. Taking a $2m-1$ order Taylor series expansion and adding the term $C_{2m}t^{2m}$.
2. Computing the Padé Approximant.
3. Computing the poles/zeroes of the Padé approximant.
4. Measuring the pole/zero locations relative to the allowed region (the disc of radius R around $t=0$).
5. Measuring the pole/zero locations relative to the previous iteration to compute dpz_i.

Steps 2 and 3 are *highly* sensitive to roundoff error in their arguments (the coefficients of the Taylor series for Step 2, and the computed coefficients of the Padé approximant for Step 3) [171, 76]. Because of the sensitive, complex nature of the dependence of Padé approximant and numerical root finding on their arguments, I have elected to use an arbitrary-precision calculation, and to increase the precision of the calculation until the result no longer changes, and use this precision as the arithmetic precision of the entire algorithm.

In this way, I ensure that the roundoff error is less than $\delta = 10^{-5}$.

Integration Error – Equation 8.22: The above algorithm uses a finite dc_i as an discrete interval to approximate the continuous probability distribution $\mathsf{Pr}(C_{2m} = c)$.

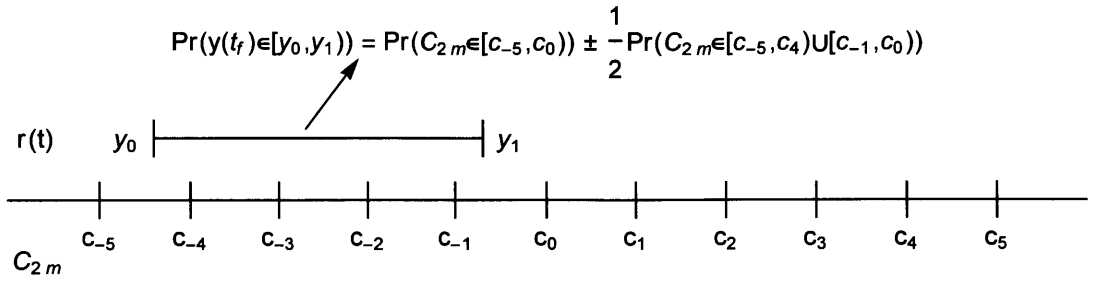


Figure 8-9: Depiction of the computation of quantization error in the process of computing $\mathsf{Pr}(y(t_f)) \in [y_0, y_1))$. The quantization error is upper bounded by taking 50% of the probability of the two ending intervals (in this case $[c_{-5}, c_{-4})$, and $[c_{-1}, c_0)$), and adding it to the overall error of the system. This is in addition to the 1% error of numerical integration computing $\mathsf{Pr}(C_{2m})$.

Moreover, the above algorithm truncates large values of c. Therefore, one must consider the error in the estimation of each interval, given by Equation 8.22. To address this concern, the algorithm in practice uses a more complex approach than shown in Figure 8-6. In Figure 8-6, dc_i is constant for all i. However, the algorithm in practice dynamically scales dc_i to keep dpz_i small in order to keep the local error tolerance of each interval below 1%. This is done using standard numerical integration techniques, recognizing that the error is bounded by the error for the rectangle rule [155]:

$$\mathrm{Err}_i = \frac{\mathrm{dc}_i^2}{2} \times \left(\frac{\mathrm{dpz}_{i+1}}{\mathrm{dc}_{i+1}} - \frac{\mathrm{dpz}_i}{\mathrm{dc}_i} \right)$$

Finally, the algorithm truncates c when dc_i reaches a maximum threshold value and dpz_i drops below the lower threshold of 10^{-4}. This corresponds to a local probability less than 10^{-5} of the maximum probability. In this way, each interval of the discrete probability distribution is within 1% of the actual probability over that interval.

Integration Error – Equation 8.23: The final integration step to compute $\mathsf{Pr}(y(t_f))$ given the first $2m-1$ derivatives of $r(t)$ is shown in Figure 8-7. To estimate the error of each interval, e.g., $\mathsf{Pr}(y(t_f)) \in [y_0, y_1)$, compute the quantization error, shown in Figure 8-9. This is then added to the 1% error of each interval of $\mathsf{Pr}(C_{2m})$.

In this way, one may compute $\mathsf{Pr}(y(t_f))$, and bounds on the numerical noise affecting its computation.

Entropy Lower Bound: First, recognize that using the above error analysis, the min-entropy of the system may be accurately computed. Because the min-entropy only considers the most probable interval of $\mathsf{Pr}(y(t_f))$, this interval always has the largest number of intervals of $\mathsf{Pr}(C_{2m})$. Therefore, the quantization error is minimized. In general, this quantization error is of the order of a few percent. This methodology is used to compute the distribution of min-entropy in Figure 8-8.

8.5 PPUF Formalism and Construction

A Public Model Physical Unclonable Function may now be formally defined. This section will provide a construction whose security reduces to Conjecture 8.4.1.

The purpose of this section is ultimately to compare the performance of an analog and digital instantiation of a computational problem. Therefore, the first step is to formalize how to transition between these domains (from continuous to discrete and vice-versa). This is done in Definitions 8.5.1 and 8.5.2.

8.5.1 Discretization of Continuous Variables

In the above analysis and in the following reduction, there are several cases where one must interpret continuous variables as discrete values. This will be done in the standard way, formalized in Definitions 8.5.1, 8.5.2.

Definition 8.5.1. *A discretization of a continuous variable y with accuracy δ, denoted $y_d \leftarrow \mathsf{D}_\delta(y)$ is the representation of y as a discrete number to precision δ. In particular, y_d is given such that:*

$$|y_d\delta - y| = \min_{i\in\mathbb{Z}} |i\delta - y| \tag{8.24}$$

This definition is extended to transforming continuous random variables into discrete random variables by integrating over the PDF.

Definition 8.5.2. *One can convert a discrete variable y_d to a continuous variable with accuracy δ, denoted $y =\!\leftarrow \mathsf{C}_\delta(y_d)$ is the representation of y_d as a continuous*

number with error $\leq \delta/2$ *by taking:*

$$y = y_d\delta + \varepsilon \tag{8.25}$$

where ε *is taken from the uniform distribution over* $[-\delta/2, \delta/2]$.

The definition is extended to transforming discrete random variables into continuous random variables by letting ε *be a random variable and summing over the PDF of* y_d.

8.5.2 Physically Accelerated Function Construction

The intent of this section is to formally construct a Physically Accelerated Function (PAF). This will be used to construct a Public Model Physically Unclonable Function (PPUF). A PAF is a combination of the physical device and a software model. Both of these instances compute a single function (cf. Definition 8.5.4). However, the physical device has the property that it obtains the result of the function faster than any digital implementation of the software model.

After formalizing the notion of a PAF (cf. Definition 8.5.5), the notion of "security" for a PAF device is formalized (cf. Definition 8.5.6). In order to do this, a "parameterized" security definition is provided (cf. Definition 8.5.7), and Lemma 8.5.14 shows a reduction of this Definition to Empirical Observation 8.4.2 for certain parameters. Conjecture 8.4.1 is used to to reduce unparameterized security (Definition 8.5.6) to parameterized security (Definition 8.5.6) in Theorem 8.5.16.

A proof sketch of the above goes as follows. Construct a "chain" of randomly chosen IVPs together (cf. Lemma 8.1.18). Show that an asymptotic improvement in solving the chain of IVPs results in an asymptotic improvement in solving an individual IVP. However, this contradicts Empirical Observation 8.4.2 in combination with Conjecture 8.4.1.

Along the way, I recognize another implication of the theory. In particular, Conjecture 8.4.1 implies a lower bound on the circuit size complexity of discrete IVP solvers, where IVPs are of the form in Definition 8.1.4.

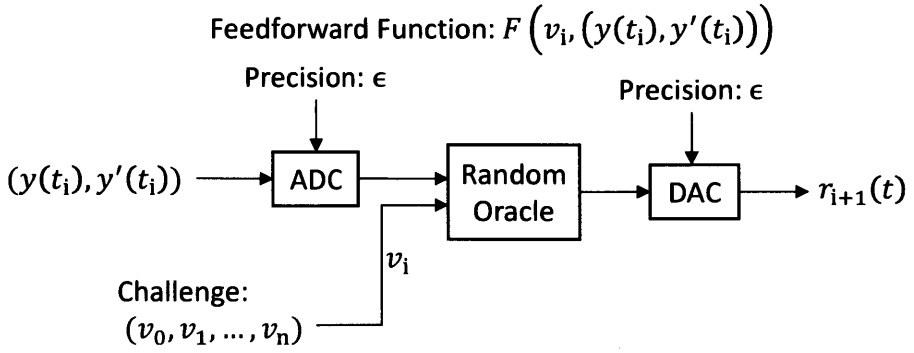


Figure 8-10: Informal depiction of the derivation function for poles/zeroes in each cluster (cf. Definition 8.5.3). The previous system state $(y(t_i), y'(t_i))$ is discretized and, alongside a piece of the challenge v_i, is sent to a random oracle. The random oracle output is then used to generate the next pole-zero cluster $r_{i+1}(t)$.

I begin with Definition 8.5.3 (informally shown in Figure 8-10), that formalizes how to derive the locations of poles/zeroes in the IVP from problem parameters.

Definition 8.5.3. *Define the function* $\mathbf{z} \leftarrow \mathsf{Fz}_{R,m,\delta}(v, M, \mathbf{y})$ *to take a portion of the challenge* $v \in \{0,1\}^l$*, the "model"* $M \in \{0,1\}^*$*, and the IVP state* $(y, y') \in \mathbb{C}^2$*. It returns a set of* m *values* $\mathbf{z} \in \mathbb{C}^m$*. (These will be used as the locations of zeroes in the IVP).*

Let $H_{R,m,\delta}(\cdot)$ *be a random oracle parameterized by* R, m, δ*, that returns* $\mathbf{z} \in \{\{0,1\}^l, \{0,1\}^l\}^m$*, wherein for all* $\{z_{\text{re}}, z_{\text{im}}\} \in \mathbf{z}$*, the following is true:*

$$(z_{\text{re}}\delta)^2 + (z_{\text{im}}\delta)^2 < R^2 \tag{8.26}$$

Define Fz *as the following algorithm:*

1: **procedure** $\mathbf{z} \leftarrow \mathsf{Fz}_{R,m,\delta}(v, M, (y, y'))$
2: Put $(y, y')_d = \mathsf{D}_\delta((y, y'))$ `// See Definition 8.5.1 for` $\mathsf{D}_\delta(\cdot)$
3: Put $\mathbf{z}_d = H_{R,m,\delta}(v, M, (y, y')_d)$
4: Put $\mathbf{z} = \mathsf{C}_\delta(\mathbf{z}_d)$ `// See Definition 8.5.2 for` $\mathsf{C}_\delta(\cdot)$
5: Return $\mathbf{z}$
6: **end procedure**

The definition is similar for $\mathbf{p} \leftarrow \mathsf{Fp}_{R,m,\delta}(v, M, (y, y'))$.

As discussed above, the function in Definition 8.5.3 will be used to "chain" IVPs

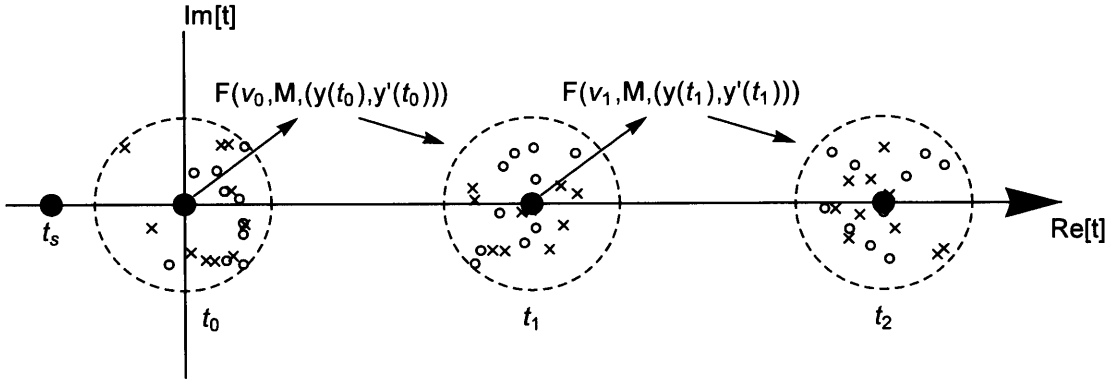


Figure 8-11: A graphical representation of the iterative numerical integration process described in Definition 8.5.4. In this case, an initial cluster of poles/zeroes around t_0 is given with radius R. The solver must numerically integrate along the real axis from some initial time t_s along the real axis to t_0. The state value at this point, $y(t_0)$, $y'(t_0)$ is then used with a random oracle to derive the locations second cluster of pole/zeroes. These are displaced by T along the path of integration. The process then repeats for n iterations.

together, by choosing the next IVP randomly. Using this function, construct the function that the PAF actually performs as an iterated numerical integration. This higher-level function is described pictorially in Figure 8-11.

Definition 8.5.4 (PAF Initial Value Problem). *A PAF Function* $(y, y') \leftarrow \mathsf{F}_{R,m,\epsilon,\delta,n}(V, M)$ *represents the mathematical function computed by the PAF device.* $(y, y') \in \mathbb{C}^2$ *represents an approximation of the output state of the ODE, a two-dimensional complex vector.* $V = \{v_i\}$ *for* i *from 0 to* $n-1$ *with* $v_i \in \{0,1\}^l$*, represents an input challenge. Finally,* $M \in \{0,1\}^*$ *represents an approximation of the model of the PAF hardware.* $(y, y') \leftarrow \mathsf{F}_{R,m,\epsilon,\delta,n}(V, M)$ *is implemented in Algorithm 4.*

The function $\mathsf{F}_{R,m,\epsilon,\delta,n}(V, M)$ from Definition 8.5.4 represents the *mathematical* function computed by the PAF and its model. It does not denote whether the function is computed in analog or digital hardware. The purpose of the PAF is to provide two computational modalities that solve the above problem. First, there is the hardware-accelerated unit, which will in general be analog in nature. Second, since the function $\mathsf{F}_{R,m,\epsilon,\delta,n}(V, M)$ is known, it can be computed by a general purpose computer for any arguments V, M by some algorithm.

Algorithm 4 Definition of the iterated initial value problem depicted in Figure 8-11.

procedure $(y, y') \leftarrow \mathsf{F}_{R,m,\epsilon,\delta,n}(V, M)$
  Set curve $C(t)$ to transit along the real axis from $-\infty \rightarrow \infty$ with constant derivative dC/dt.
  Choose $T \gg R$. `// Separate each problem by` $2T \gg R$
  Set $t = -T$. `// Start Integrating IVP from` $-T$.
  Set $y(-T) = 1$, $y'(T) = 0$. `// Initialize IVP`
  Set $\mathbf{y}_{\text{meas}} = \{y(-T), y'(-T)\}$.
  for i from 1 to n **do**
    Construct $\mathbf{z} = \mathsf{Fz}_{R,m,\delta}(v_i, M, \mathbf{y}_{\text{meas}})$. `// Calculate Poles and Zeroes, construct ODE`
    Construct $\mathbf{p} = \mathsf{Fp}_{R,m,\delta}(v_i, M, \mathbf{y}_{\text{meas}})$.
    For all $z \in \mathbf{z}$, set $z = z + T$
    For all $p \in \mathbf{p}$, set $p = p + T$
    Construct $L = \partial_t^2 + r(t)$ with $r(t)$ poles/zeroes at p, z respectively.
    Set $\mathbf{IC} = \{y(t), y'(t)\}$.
    Construct $\mathbf{IVP} = \{L, C(x), t, \mathbf{IC}, \epsilon\}$
    Integrate **IVP** from t to $t + T$. `// Integrate into the center of the pole/zero cluster`
    Set $\mathbf{y}_{\text{meas}} = \{y(t+T), y'(t+T)\}$. `// Store state` $\mathbf{y}_{\text{meas}}$ `for use to compute next stage`
    Integrate **IVP** from $t + T$ to $t + 2T$.`// Integrate out of the cluster of poles/zeroes`
    Set $t = t + 2T$.
  end for
  return $\{y(t), y'(t)\}$.
end procedure

These two devices (the hardware accelerator, and the software algorithm), comprise the PAF, which is formalized in Definition 8.5.5.

Definition 8.5.5 (Physically Accelerated Function). *A* $(\mathsf{F}_{R,m,\epsilon,\delta,n}, l, \chi)$*-Physically Accelerated Function is a pair of objects representing a function* $(y, y') \leftarrow \mathsf{F}_{R,m,\epsilon,\delta,n}(V, M)$*, where* $V = \{v_i\}$ *for* i *from 0 to* $n-1$. v_i, $M \in \{0,1\}^l$. (y, y') *are continuous/discrete values related through Definitions 8.5.1 and 8.5.2. The value* V *is the "challenge" of the system, chosen at random for each call to the PAF. The value* M *is the "model," which is constant.*

- *The "PAF Device," termed* $(y, y') \leftarrow \mathrm{PAF}_M(V)$*, that is a specialized piece of hardware that computes the function* $(y, y') \leftarrow \mathsf{F}_{R,m,\epsilon,\delta,n}(V, M)$.
- *A model* $M \in \{0,1\}^*$ *of* PAF_M*, and a digital algorithm* $(y, y') \leftarrow \mathsf{SWF}(V, M)$ *that computes the function* $(y, y') \leftarrow \mathsf{F}_{R,m,\epsilon,\delta,n}(V, M)$.

The two objects PPAF_M *and* SWF *have the property that* $\forall V \; \|\mathsf{SWF}(V, M) - \mathrm{PAF}_M(V)\| < \epsilon$.

For the purpose of this analysis, the PAF model M is constant, publicly known. It won't be used in the security reduction that follows. The model will become important when defining a PPUF later.

Define the security of the above Physically Accelerated Function in terms of the experiment in Definition 8.5.6.

Definition 8.5.6 (PAF Security). *A* $(\mathsf{F}_{R,m,\epsilon,\delta,n}, l, \chi)$*-PAF is* ϕ*-secure with error* p_{err} *and noise* ϵ *if the following is true:*

First, the correctness condition must be satisfied:

$$\mathsf{Pr}_V \left(\|\mathrm{PAF}_M(V) - \mathsf{SWF}(V, M)\| > \epsilon \right) < p_{\text{err}}$$

Second, the soundness condition must be satisfied: for all PPT A, $\mathsf{Adv}_{\mathsf{M}}^{\mathrm{t-uprd}}(\mathsf{A}) < \phi$*, which is defined in terms of the following timed-unpredictability* $(\mathrm{t-uprd})$ *experiment.*

1: **procedure** $\{0,1\} \leftarrow \mathsf{Exp}_{\mathsf{M}}^{\mathrm{t-uprd}}(\mathsf{A})$

2: Choose random V.

3: Run $(y_{\mathrm{PAF}}, y'_{\mathrm{PAF}}) \leftarrow \mathrm{PAF}_M(V)$, measure time t_{PAF}.

4: Run $(y_{\mathsf{A}}, y'_{\mathsf{A}}) \leftarrow \mathsf{A}(V, M)$, measure time t_{A}.

5: **if** $|y_{\mathrm{PAF}} - y_{\mathsf{A}}| < \epsilon$ and $t_{\mathrm{PAF}} \geq t_{\mathsf{A}}$ **then return** 1

6: **else return** 0

7: **end if**

8: **end procedure**

The t − uprd *advantage of* A *is defined as*

$$\mathsf{Adv}_{\mathsf{M}}^{\mathrm{t-uprd}}(\mathsf{A}) = \mathsf{Pr}\left(\mathsf{Exp}_{\mathsf{M}}^{\mathrm{t-uprd}}(\mathsf{A}) = 1\right)$$

8.5.3 Security Reduction

There are two primary results in this section. First, Theorem 8.5.13 show that there exists a class of IVPs that are "n-complex" (Definition 8.5.12), which informally requires at least n calls to an expansion algorithm when the numerical integrator is of the form of Definition 8.3.3. This implies an asymptotic lower bound on the circuit size complexity of any numeric integration method for this class of IVPs.

The second result is the security reduction of a PAF construction. Definition 8.5.6 provides the formal definition for PAF security. However, there are several steps to reduce the above definition to the cryptographic assumptions, a security parameter, and the physical properties of the PAF_M relative to the software model SWF. This is done by using the chaining function from Definition 8.5.3 to show that a digital algorithm violating a parameterized security definition (cf. Definition 8.5.7) by being asymptotically faster than a PAF that computes a function of the form in Definition 8.5.4 can be used to accelerate the computation of random IVPs from $U_{R,m,\delta}$. This contradicts Empirical Observation 8.4.2. Further, Conjecture 8.4.1 is used to tie this result to the primary security Definition 8.5.6, culminating in Theorem 8.5.16.

Therefore, the first step is to provide a "parameterized" version of the PAF security

game in Definition 8.5.6. This is formalized in Definition 8.5.7 using the "maximum-order expansion unpredictability" (moe – uprd) experiment described below. Note that Definition 8.5.7 requires an adversary of the form described in Conjecture 8.4.1.

Definition 8.5.7 (Parameterized PAF Security). *A* $(\mathsf{F}_{R,m,\epsilon,\delta,n}, l, \chi)$*-PAF is* ϕ*-secure with error* p_{err}*, noise* ϵ *to order* k *for* n *iterations if the following is true:*

First, the correctness condition must be satisfied:

$$\mathsf{Pr}_V \left(\|\text{PAF}_M(V) - \mathsf{SWF}(V, M)\| > \epsilon \right) < p_{\text{err}}$$

Second, the soundness condition must be satisfied: for all PPT numerical integration algorithms (cf. Definition 8.3.3) A, $\mathsf{Adv}_{\mathsf{M}}^{\text{moe-uprd}}(\mathsf{A}) < \phi$*, which is defined in terms of Equation 8.27.*

Let A *be comprised of an expansion algorithm family indexed by expansion order* k*, and denoted as* $\mathsf{Exp}^{\mathsf{A}}{}_k$ *as well as a global numerical integration* NI^{A} *(cf. Definition 8.3.3). This expansion algorithm family may be single or multi-step as in Definitions 8.3.1 and 8.3.2. Both* $\mathsf{Exp}^{\mathsf{A}}{}_k$*, and* NI^{A} *may have access to the random oracle* $H(\cdot)$*.*

1: **procedure** $\{0,1\} \leftarrow \mathsf{Exp}_{\mathsf{M}}^{\text{moe-uprd}}(\mathsf{A})$
2: Choose random V.
3: Run $r \leftarrow \text{PAF}_M(V)$.
4: Run $r' \leftarrow \mathsf{A}(V, M)$, measure N number of stages of NI (cf. Figure 8-3), requiring maximum expansion order $k_{\max}$ during each stage.
5: **if** $|r' - r| < \epsilon$ and $N < n$ and $k_{\max} < k$ **then return** 1
6: **else return** 0
7: **end if**
8: **end procedure**

The moe – uprd *advantage of* A *is defined as*

$$\mathsf{Adv}_{\mathsf{M}}^{\text{moe-uprd}}(\mathsf{A}) = \mathsf{Pr}\left(\mathsf{Exp}_{\mathsf{M}}^{\text{moe-uprd}}(\mathsf{A}) = 1\right) \tag{8.27}$$

To start, the PAF from Definition 8.5.5 is already a PAF for 1 iteration of $\mathsf{F}_{R,m,\epsilon,\delta,n=1}$ (i.e., $n = 1$). That is, any software model of the PAF that satisfies correctness will require > 1 expansions with some non-negligible probability in m to compute the output r. Intuitively, this is because there is still entropy remaining in $y(t_f)$ after only a single expansion has been computed. Therefore, the algorithm cannot guess the output (y, y') with high probability. This is formalized in Lemma 8.5.8.

Lemma 8.5.8. *Put $n = 1$. Given $\mathsf{F}_{R,m,\epsilon,\delta,n=1}$, there exists a choice of R, m, ϵ, T such that the following is true:*

Definition 8.5.5 of a $(\mathsf{F}_{R,m,\epsilon,\delta,n}, l, \chi)$-PAF is $(\phi = 1/\mathsf{poly}(m))$-secure with error p_{err}, noise ϵ to order $k = 2m - 1$ for $n = 1$ iterations.

Proof. With $n = 1$ iterations, the response is computed by numerically integrating from $t_s < -R$ to $t_f = 0$. Given a uniformly random challenge V, the IVP is uniformly chosen from $U_{R,m,\delta} = \{\mathbf{IVP}_{\epsilon,C(x),t_s}, \mathbf{Rslt}_{t_f}\}$ as in Definition 8.2.2.

By Empirical Observation 8.4.2, there exists a choice of R, m, ϵ, T such that an algorithm A that is of the form from Definition 8.3.3 with maximum expansion order $k_{\max} \leq 2m - 1$ that uses ≤ 1 calls to $\mathsf{Exp}^{\mathsf{A}}{}_k$ to integrate $y(t)$ from t_s to t_f with error ϵ cannot succeed with greater than $1/2$ probability for a fraction of challenges non-negligible in m. This is because Empirical Observation 8.4.2 states that for a fraction of challenges non-negligible in m, there is greater than 1 bit of entropy remaining in $y(t_f)$ after a single expansion of maximum order $2m - 1$ at t_s.

This follows from the fact that with non-negligible probability in m, there is at least one bit of entropy left in $y(t_1)$, when $y(t_1)$ is measured to precision ϵ, when only the first $2m - 1$ derivatives of $r(t)$ are revealed to $\mathsf{Exp}^{\mathsf{A}}{}_k$. Therefore, $\mathsf{Exp}^{\mathsf{A}}{}_k$ does not have enough information to compute $y(t_1)$ to precision $\leq \epsilon$, regardless of its runtime.

Note that since Empirical Observation is constructive in terms of its choice of R, m, ϵ, T, these parameters may be used in the above secure PAF construction.

□

Furthermore, in Lemma 8.5.9 that the above property holds for $n = 1$, even in

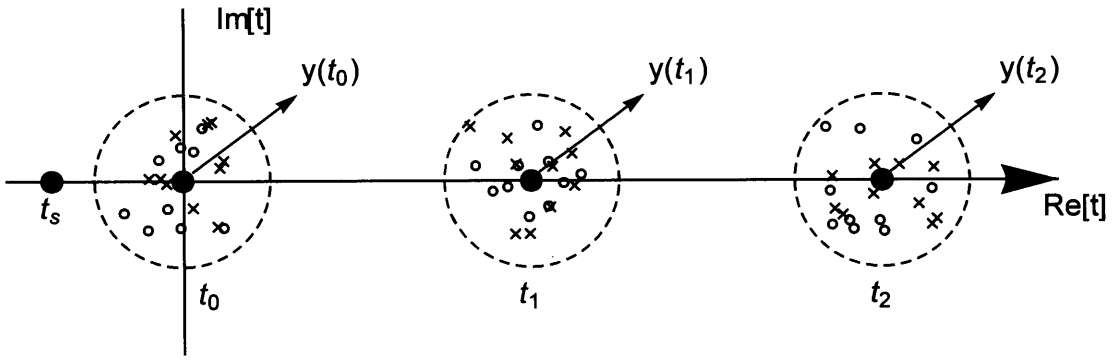


Figure 8-12: A depiction of F^{new} from Observation 8.5.9. Instead of defining each cluster of poles and zeroes based on the system state at a previous time (as in Figure 8-11, Definition 8.5.4), one may consider each pole/zero cluster as statically chosen. The problem is then to compute the system state $(y(t), y'(t))$ at *all* points $t = \{t_0, t_1, t_2, ...\}$, instead of just $t = t_f$.

the presence of other pole-zero clusters far away from the path of integration from t_s to t_f. In essence, adding another cluster of pole-zero pairs far away from the initial cluster does not affect the difficulty of the problem.

Lemma 8.5.9. *Consider a* $(\mathsf{F}_{R,m,\epsilon,\delta,n}, l, \chi)$*-PAF that is* $(\phi = 1/2)$*-secure with error* p_{err}*, noise* ϵ *to order* $k = 2m - 1$ *for* $n = 1$ *iterations.*

By definition $\mathsf{F}_{R,m,\epsilon,\delta,n}$ *is defined in terms of the ODE* $y'' = r(t)y(t)$*, where* $r(t)$ *contains* m *poles,* m *zeroes on the disc of radius* R *around* $t = 0$*.*

The following $(\mathsf{F}^{\text{new}}{}_{R,m,\epsilon,\delta,n}, l, \chi)$*-PAF has the same security as the original PAF, where* $\mathsf{F}^{\text{new}}{}_{R,m,\epsilon,\delta,n}$ *is defined in terms of the ODE* $(y^{\text{new}})'' = r(t)r^{\text{new}}(t)y^{\text{new}}(t)$*, with* $r^{\text{new}}(t)$ *has* m *poles,* m *zeroes on the disc of radius* R *around* $t = T$*, for some finite* $T \gg R$*.*

Proof. This follows from the fact that $r(t)r^{\text{new}}(t)$ is analytic along the curve of integration $C(x)$, and therefore $y^{\text{new}}(t)$ is also analytic along $C(x)$.

Recognize that $r^{\text{new}}(t)$ will be of the form:

$$r^{\text{new}}(t) = r_{\text{lim}} \times \frac{\prod_{i=1}^{m_z} 1 - \frac{t-T}{z_i - T}}{\prod_{i=1}^{m_p} 1 - \frac{t-T}{p_i - T}}$$

One may compute a series of this expression in the limit ($T \gg R \gg t$), which results in:

$$r^{\text{new}}(t) = 1 + O(t/T)$$

Therefore, in the limit of large T, the expression $r(t)r^{\text{new}}(t) \to r(t)$. Therefore, $y^{\text{new}}(t) \to y(t)$ in the limit of large T as well. For sufficiently large T, $\|y^{\text{new}}(t) - y(t)\| < \epsilon$. An algorithm that has advantage in computing $y^{\text{new}}(t_f)$ in such a problem could therefore be used to gain advantage in the original problem of computing $y(t_f)$, since they are less than ϵ apart.

In conclusion, if a $(\mathsf{F}_{R,m,\epsilon,\delta,n}, l, \chi)$-PAF is secure by Definition 8.5.7, then there exists some T for which $(\mathsf{F}^{\text{new}}{}_{R,m,\epsilon,\delta,n}, l, \chi)$-PAF is also secure.

□

The above proof was in the case of a single additional cluster of pole-zero pairs at a distance T away from the first cluster. This can be trivially extended to multiple clusters by the same argument.

Corollary 8.5.10. *Consider a $(\mathsf{F}_{R,m,\epsilon,\delta,n}, l, \chi)$-PAF that is $(\phi = 1/2)$-secure with error p_{err}, noise ϵ to order $k = 2m - 1$ for $n = 1$ iterations.*

By definition $\mathsf{F}_{R,m,\epsilon,\delta,n}$ is defined in terms of the ODE $y'' = r(t)y(t)$, where $r(t)$ contains m poles, m zeroes on the disc of radius R around $t = 0$.

The following $(\mathsf{F}^{\text{new}}{}_{R,m,\epsilon,\delta,n}, l, \chi)$-PAF has the same security as the original PAF, where $\mathsf{F}^{\text{new}}{}_{R,m,\epsilon,\delta,n}$ is defined in terms of the ODE $(y^{\text{new}})'' = r(t)r_1^{\text{new}}(t)r_2^{\text{new}}(t)\cdots y^{\text{new}}(t)$, with $r_i^{\text{new}}(t)$ has m poles, m zeroes on the disc of radius R around $t = T$, for some finite $T \gg R$.

Proof. Using the same argument as above, recognize that in the limit of large T:

$$\lim_{|T|\to\infty} r_i^{\text{new}}(t) = 1$$

Therefore, for any finite n, where i from 1 to n, there exists a large enough T such that $\|y^{\text{new}}(t) - y(t)\| < \epsilon$. In conclusion, if a $(\mathsf{F}_{R,m,\epsilon,\delta,n}, l, \chi)$-PAF is secure by

Definition 8.5.7, then there exists some T for which $(\mathsf{F}^{\mathsf{new}}{}_{R,m,\epsilon,\delta,n}, l, \chi)$-PAF is also secure.

□

Since the addition of n pole-zero clusters far away from the IVP in question does not affect its difficulty, Theorem 8.5.13 shows that solving *all* of these problems simultaneously requires greater than n expansions, which provides a lower-bound on the circuit complexity of solving the IVP. I begin by providing a definition of these n pole-zero cluster IVPs in Definition 8.5.11.

Definition 8.5.11. *The family of n-cluster IVPs, $S_{n,T,U}$ is a family of IVPs, wherein n samples u_j (j from 1 to n) are taken from $U_{R,m,\delta}$ are taken. Each is then displaced (Definition 8.1.15) by jT (along the real axis), and a curve of integration is taken from $t_s < -R$ through $t_j = jT$.*

The solution consists of $\{(y(t_j), y'(t_j)\}$, the state at each t_j.

Using the definition of this n-cluster, Definition 8.5.12 states that any algorithm requires at least n expansions in order to compute $y(t_j)$ for all j from 1 to n. Informally, this is the "security game," that the first primary result is derived from. That is, this definition is used to show that if Conjecture 8.4.1 is true, then there exists an IVP that is n-complex.

Definition 8.5.12. *Define a family of n-cluster IVPs is "n-complex with noise ϵ, error probability p_{err} to order $k_{\max}$" if for all PPT numerical integration algorithms A of the form in Definition 8.3.3, the following is true:*

Let $\{(y_{\text{A}}, y'_{\text{A}})_j\} \leftarrow \mathsf{A}(s)$ with j from 1 to n.

If the algorithm is correct: $\mathsf{Pr}_{s \in S_{n,T,U}} (\|(y_s(t_j), y'_s(t_j)) - (y_{\text{A}}, y'_{\text{A}})_j\| > \epsilon) < p_{\text{err}}$ for all j, then it is true that A requires $> n$ calls to $\mathsf{Exp}^{\mathsf{A}}{}_k$ and has maximum expansion order $k_{\max}$.

Theorem 8.5.13 now reduces the above definition to Conjecture 8.4.1 and Empirical Observation 8.4.2. A Corollary of this Theorem is that there exists an asymptotic lower bound on circuit size for any digital circuit that solves problems from $S_{n,T,U}$.

Theorem 8.5.13. *Let $U_{R,m,\delta}$ be chosen such that Empirical Observation 8.4.2 holds.*

An n-cluster IVP family $S_{n,T,U}$ is n-complex with noise ϵ, error probability p_{err} to order $k_{\max}$, for some choice of T.

Proof. Consider n IVPs u_j sampled randomly from $U_{\text{R,m},\delta}$. By Empirical Observation 8.4.2, each u_j individually has the property that after 1 expansion at t_s of order at most $2m-1$, there is a non-negligible amount of entropy remaining in $\mathsf{Pr}(y(t_f))$.

By Lemma 8.5.9, for sufficiently large displacement T, one may combine these individual u_j into a single IVP (cf. Definition 8.5.11) that is a member of $S_{n,T,U}$. Furthermore, the combination of these IVPs does not affect their individual property that there remains a non-negligible amount of entropy in $\mathsf{Pr}(y(t_j))$ after the computation of a single expansion up to order $2m-1$ at some t_s.

Therefore, consider an adversary A that requires $\leq n$ calls to $\mathsf{Exp}^{\mathsf{A}}{}_k$ to achieve the correctness condition stated in the Theorem with $p_{\text{err}} < 1 - \mathsf{neg}(n)$. Construct an algorithm B that takes n random instances of IVPs from the family $U_{R,m,\delta}$ as follows.

Let u_j be n unique samples from $U_{\text{R,m},\delta}$. From Lemma 8.1.18, connect these IVPs into a single IVP that is identical to an element of $S_{n,T,U}$. Using A, solve this problem using $\leq n$ expansions to compute the system state $(y(t_j), y'(t_j))$ at each t_j at the center of each cluster, with $p_{\text{err}} < 1 - \mathsf{neg}(n)$.

Therefore, B requires (amortized) ≤ 1 expansions per random IVP instance from the family $U_{R,m,\delta}$. Moreover, recognize that since each random IVP instance u_j has the property that after one expansion, there is a non-negligible amount of entropy remaining in $\mathsf{Pr}(y(t_j))$ that is *independent* of n (i.e., $h(y(t_j)) = 1/\mathsf{poly}(m) = \Theta(n^0)$), the joint entropy of each of these systems is $\sum_{j=1}^{n} h(y(t_j)) = \Theta(n)$. Therefore, because $p_{\text{err}} < 1 - \mathsf{neg}(n)$, there must be at least one u_j, where A has computed $\mathsf{Pr}(y(t_j))$ with entropy $o(n^0)$.

This contradicts Empirical Observation 8.4.2.

□

Note that Theorem 8.5.13 shows how an IVP can require at least n expansions in order to compute n unique points in the IVP. This provides a non-trivial circuit

lower bound. By the definition of the problem, any IVP with $\Theta(n)$ parameters (e.g., pole-zero locations for the system in question) necessarily has a solver with circuit size of $\Theta(n)$ because there are $\Theta(n)$ inputs. However, Theorem 8.5.13 shows that any discrete solver must have circuit size $\Omega(n \times \mathsf{Size}(\mathsf{Exp}^{\mathsf{A}})$, where $\mathsf{Size}(\mathsf{Exp}^{\mathsf{A}})$ is the circuit size of an expansion algorithm.

I now move on to the second primary result of this section. In particular, Theorem 8.5.16 proves that Conjecture 8.4.1 in combination with Empirical Observation 8.4.2 implies the existence of a PAF that is secure according to Definition 8.5.6.

I begin by providing a reduction of the PAF security according to Definition 8.5.7 to Empirical Observation 8.4.2 in Lemma 8.5.14. Theorem 8.5.16 uses Conjecture 8.4.1 and a required property of the system (Definition 8.5.15) to show PAF security according to Definition 8.5.6.

Lemma 8.5.14. *Let $U_{R,m,\delta}$ be chosen such that Empirical Observation 8.4.2 holds, and let $\mathsf{F}_{R,m,\epsilon,\delta,n}$ sample uniformly from $U_{R,m,\delta}$.*

There exists a choice of ϵ such that a $(\mathsf{F}_{R,m,\epsilon,\delta,n}, l, \chi)$-PAF is ϕ-secure with error p_{err}, noise ϵ to order $k \leq 2m-1$ for n iterations, where $\phi = \mathsf{neg}(n)$, and $p_{\text{err}} < 1 - \mathsf{neg}(n)$.

Proof. First recognize that the configuration of the $i+1$'th pole-zero cluster depends via a random oracle on the system state at the center of the i'th pole-zero cluster (cf. Definition 8.5.3).

As a result, the adversary cannot guess the configuration of the second pole-zero cluster. Therefore, stage 1 of the adversary (cf. Figure 8-3) can only integrate through the first cluster. This is shown in Figure 8-13.

Now, recognize that if the number of stages of the adversary N is less than the number of clusters n, then there exists some stage i that integrates through at least 2 clusters. This is shown in Figure 8-14.

However, for the i'th stage to have performed this integration, it must have known the configuration of the $i+1$'th pole-zero cluster. For it to have known the configuration of the cluster, it must have already (at the beginning of the stage), known the

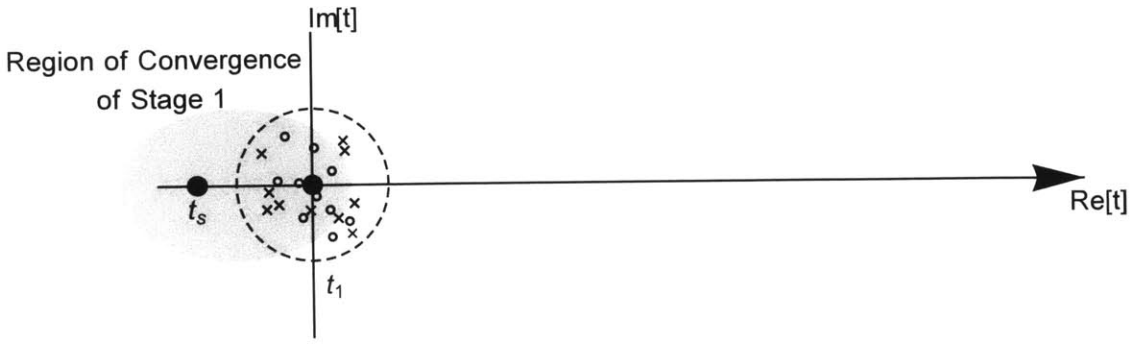


Figure 8-13: Base case of the PAF adversary. Because the adversary cannot know the configuration of the second cluster of pole-zero pairs, it can only integrate through the first cluster. Note that although Empirical Observation 8.4.2 states that a single expansion cannot have a convergence region as large as depicted above, multiple expansions computed in parallel may be combined to obtain a convergence region this large.

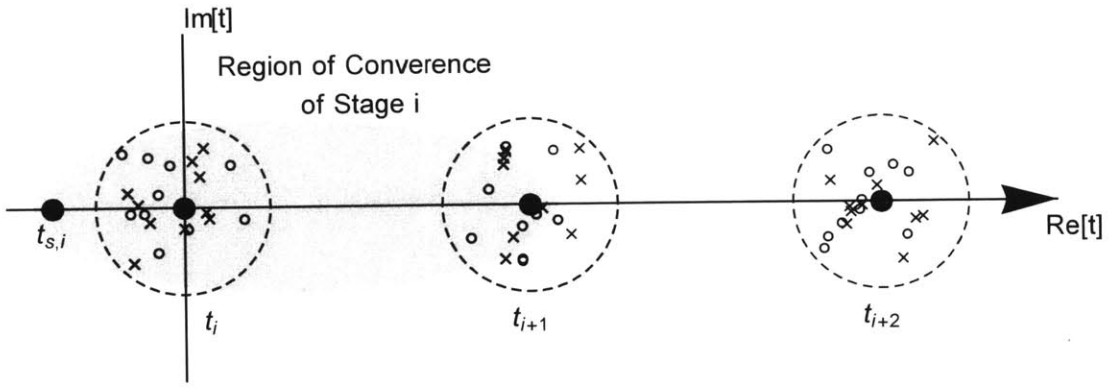


Figure 8-14: Inductive case of the PAF adversary. If $N < n$, then there is some $i < N$ that integrates through 2 clusters.

system state at the center of the i'th cluster (since the configuration of the $i+1$'th pole-zero cluster is related through a random oracle).

This implies that the i'th stage did not integrate over two clusters, rather the $i-1$'th stage had already integrated over two clusters. Repeat the inductive argument until $i = 1$. This contradicts the base case. □

The above Lemma 8.5.14 shows that the parameters from Empirical Observation 8.4.2 may be used to construct a PAF that is secure under the parameterized Definition 8.5.7 using the construction in Definition 8.5.5, using the function from Definition 8.5.4.

The next and final step is to relate this security back to the original definition of PAF security, Definition 8.5.6. In order to do this, there are two pieces needed. First, use Conjecture 8.4.1 to show that any algorithm that could violate the security of Definition 8.5.6 would also be a valid algorithm that violates Definition 8.5.7.

The second piece is to relate the runtime of the software model to the number of calls to $\mathsf{Exp}^{\mathsf{A}}{}_k$. To start, I claim that $\mathsf{Exp}^{\mathsf{A}}{}_k$ has runtime that is $\Theta(1)$. This directly implies from Lemma 8.5.14 that any algorithm will have runtime $\Omega(n)$, since at least n calls to $\mathsf{Exp}^{\mathsf{A}}{}_k$ are needed.

By definition, the analog computer runs in $\Theta(n)$ time. Therefore, at worst, the two computational modalities ($\mathsf{SWF}(V, M)$ and $\mathsf{PAF}_M(V)$) are related by a constant factor.

This constant factor is the purpose of Definition 8.5.15. It defines a condition in which the physical hardware in PAF_M is a constant-factor faster than a minimal element of $\mathsf{Exp}^{\mathsf{A}}{}_k$.

Note that this definition *is not constructive* – it does not claim that such a device exists. However, there is reason to believe with best-known implementations of technology that it does. This is discussed further in Chapter 10.

Assuming the existence of such a device, I use it to construct a PAF that is secure according to Definition 8.5.6 in Theorem 8.5.16.

Definition 8.5.15. *An* $(\mathsf{F}_{R,m,\epsilon,\delta,n}, l, \chi)$*-PAF is* θ*-hardware accelerated over an implementation of* $\mathsf{Exp}^{\mathsf{A}}{}_k$ *if* $\theta \times \mathsf{Runtime}(\mathsf{PAF}_M(V)) \leq \mathsf{Runtime}(\mathsf{Exp}^{\mathsf{A}}{}_k)$ *for all* k*.*

Informally, Definition 8.5.15 defines a θ parameter which defines the "constant-factor" between PAF_M and SWF. A θ parameter greater than 1 implies that there is some constant-factor advantage that PAF_M has over the computational modality implementing SWF.

Theorem 8.5.16. *Let* $U_{R,m,\delta}$ *be chosen such that Empirical Observation 8.4.2 holds, and let* $\mathsf{F}_{R,m,\epsilon,\delta,n}$ *choose from* $U_{R,m,\delta}$*.*

Let a $(\mathsf{F}_{R,m,\epsilon,\delta,n}, l, \chi)$*-PAF be* ϕ*-secure with error* p_{err}*, noise* ϵ *to order* k *for* n *iterations, where* $\phi = \mathsf{neg}(n)$ *according to Definition 8.5.7.*

If $(\mathsf{F}_{R,m,\epsilon,\delta,n}, l, \chi)$*-PAF is* θ*-hardware accelerated against any adversary* A *with* $\theta > 1$*, then the* $(\mathsf{F}_{R,m,\epsilon,\delta,n}, l, \chi)$*-PAF is* ϕ*-secure with error* p_{err} *and noise* ϵ *according to Definition 8.5.6.*

Proof. First, recognize that if a $(\mathsf{F}_{R,m,\epsilon,\delta,n}, l, \chi)$-PAF is θ-hardware accelerated for $\theta > 1$, then the adversary A can perform at most n/θ calls to $\mathsf{Exp}^{\mathsf{A}}{}_k$. Therefore, A must make $\leq n$ calls to $\mathsf{Exp}^{\mathsf{A}}{}_k$.

Therefore, by Conjecture 8.4.1, any algorithm that correctly solves for a subset of $U_{R,m,\delta}$ must be of the form in Definition 8.3.3. This implies that an algorithm A that has non-negligible $\mathsf{Adv}_M^{\text{t-uprd}}(\mathsf{A})$ in Definition 8.5.6 must also have non-negligible $\text{Adv}_M^{\text{moe-uprd}}(\mathsf{A})$ in Definition 8.5.7.

This in turn contradicts Lemma 8.5.14.

□

8.5.4 Proof Remarks

The Kovacic algorithm is not directly invoked in this reduction. It is assumed that the algorithm is run in advance to ensure that the ODE does not have a closed form solution, so that Conjecture 8.4.1 may be applied. Empirically, it is virtually always the case that the ODE does not have a closed-form solution.

The form of the adversary in Definition 8.3.3 (also Figure 8-3), allows for parallelism in each stage, and does not include the computational cost of combining the individually computed expansions in each stage.

Need for a Random Oracle: A random oracle is required to derive the configuration of each cluster in order to enforce that regardless of the input distribution, the output distribution looks uniform (assuming sufficient min-entropy of the input). This is important, as Conjecture 8.4.1 applies only to the family of ODEs in Definition 8.2.2, which has a uniform distribution of poles and zeroes. If the adversary has some incomplete knowledge of the distribution of $y(t_i)$, and this impacts the distribution of the next pole-zero cluster, then this could conceivably result in a computational advantage for the adversary.

Hybrid Analog-Digital System: By using the random oracle above, the PAF becomes a hybrid analog-digital computer. The implications of this requirement are discussed more in the context of possible PAF architectures in Chapter 10. In general, the conversion between analog/digital domains introduces additional overhead in the analog computer. The speedup of the analog computer over the digital computer must remain even in the presence of this additional overhead. Namely, Definition 8.5.15 includes the *total runtime* of PAF_M, including the delay introduced in the ADC/DAC used in the procedure from Definition 8.5.3.

Further note that the random oracle does not have any speedup in the PAF versus SWF. Therefore, it is required that the random oracle implementation in the PAF be efficiently implemented such that the SWF cannot achieve a speedup over PAF through more efficient computation of the random oracle.

Min-entropy of System State: Even with the use of a random oracle, there needs to be sufficient min-entropy input from the system state $y(t_i)$ in order to (1) have the probability distribution look uniform, and (2) prevent possible parallelization of the solution. However, because the ODE ultimately must be realized in a physical system, there are physical limitations on the entropy that may be obtained in the system state. In particular, if $y(t)$ is an analog value, measurement precision of the analog system is generally not more than 10-20 bits. (This is discussed in greater

detail in Chapter 10.) This results in a best-case number of configurations in each stage on the order of 2^{10} to 2^{20}, which is small enough to be solved in parallel.

This may be dealt with by running multiple analog computers in parallel. This is not depicted in the reduction, but is a simple extension to the protocol. The system state then is the concatenation of the states of each analog computer (i.e., if the system state of each analog computer is 10 bits, then 13 analog computers running in parallel would have an overall system state of 130). The overall system state (130 bits in the above example) is then fed into each of the random oracles deriving the next cluster for each analog computer. Each analog computer receives a unique challenge (which is concatenated with the system state prior to running the random oracle), so the configurations for each analog computer are relatively random.

Noise of the Analog Computer: In the above reduction, it is assumed that the analog computer has noise $\leq \epsilon$. That is, the ideal system state $(y(t), y'(t))$ and the analog computer's state $(\hat{y}(t), \hat{y}'(t))$ are at most ϵ apart in state space (i.e., $(y(t) - \hat{y}(t))^2 + (y'(t) - \hat{y}'(t))^2 \leq \epsilon^2$).

If ϵ is too small, then this property does not hold, and there is some probability of error in the analog computation in each cluster. This probability of success decays exponentially per stage to zero. Informally, this may be seen as chaotic behavior of the system. Although each ODE is linear (and therefore not chaotic), the feed-forward random oracle introduces nonlinearity. The feed-forward mechanism is a discretization followed by a random oracle (cf. Definition 8.5.3). The discretization step corrects errors, and the random oracle amplifies them. If ϵ is too small, then the discretization step does not correct all errors, and the remaining error is amplified to full scale by the random oracle, resulting in chaotic behavior that cannot be simulated.

On the other hand, if ϵ is too large, then Empirical Observation 8.4.2 no longer holds, and the reduction breaks down.

8.5.5 PPUF Construction

The PAF formalism used above already contains the required parameters to enable a PPUF system. The PAF has a "model" parameter M, which is described as a bit

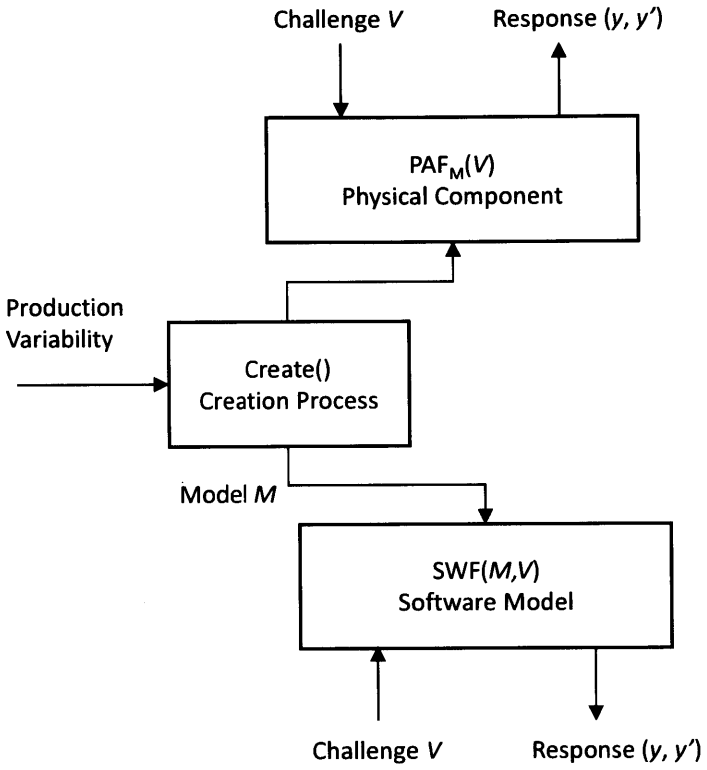


Figure 8-15: A diagram of the formal PPUF construction using a PAF. Note that the model M that is passed to the software function, and that parameterizes the physical component is distributed according to some distribution χ. Because M is used as part of an argument to a random oracle in this model (cf. Definition 8.5.3), it is only required that M have sufficient min-entropy.

string $\{0,1\}^l$. It is unused in the PAF speedup reduction, and is therefore set to a constant.

If instead this parameter is a random variable distributed according to some distribution χ, then the unclonability property can be achieved if certain properties on χ are met. Note that this follows the approach taken by the standard PUF formalism [10]. Figure 8-15 depicts the PPUF creation and challenge/response process when the model M is a random variable. Note that this figure analogizes to the "physical component" piece of the standard PUF formalism described in [10].

In the PAF formalism, the model M is used as an argument to the random oracle $H_{R,m,\delta}$. Therefore, as long as M has sufficient min-entropy, the likelihood of finding two PPUF clones is minimized. This is the only requirement on M.

8.5.6 "Security Parameter" for PPUF Systems

An interesting observation of the PPUF formalism above is the new type of "security parameter." In a traditional cryptosystem, a security parameter of k corresponds to a minimum number of bit-operations of 2^k to be performed by an adversary in order to break the algorithm.

In the above formalism, the "security parameter" corresponds to the maximum expansion order k that can be feasibly computed by an adversary. That is, the above algorithm is secure until an adversary is capable of computing an expansion of order $> k$ in time less than $\mathsf{Runtime}(\mathrm{PPUF}_M(V))$, the overall runtime of the PPUF hardware.

This is unique for two reasons.

- The circuit complexity is not reduced to bit-operations, rather it is reduced to the notion of an "expansion", which is at its core an arithmetic operation (multiply and/or add operations) and therefore more complex than a single bit operation.

- It provides a time-limit on the computation based on the PPUF hardware, $\mathrm{PPUF}_M(V)$. The difference in time between the PPUF hardware and the runtime for $\mathsf{Exp}^{\mathsf{A}}{}_k$ serves as a bound for the constant-factor speedup of the PPUF over the model.

9 - Discussion of the Primary Conjecture

The purpose of this chapter is to provide justification, analysis, and discussion for Conjecture 8.4.1. I first discuss the various methods of ODE approximation in the context of their application, concluding that *asymptotic approximation* is the only feasible approach for studying initial value problems (the problems used in the PAF in Chapter 8). Asymptotic expansions are by their definition of the form described in Definition 8.3.3.

Second, I focus on a justification for a finite expansion order and intuition for Empirical Observation 8.4.2 by providing a construction of an asymptotic expansion algorithm for the ODEs described in Chapter 8. This expansion explicitly demonstrates how to leverage the structure of the ODE to achieve improved performance over existing expansions (discussed non-constructively in Chapter 8. Finally, it is recognized that the cost of any expansion grows with expansion order due to several factors (numerical precision, number of arithmetic operations, etc.). It is observed that due to these limitations, most practical numerical integration algorithms are of relatively low order (less than order 10). This completes the justification for Conjecture 8.4.1. Further, by providing an explicit construction, I demonstrate concretely the principles discussed in Chapter 8 in the analysis of Empirical Observation 8.4.2.

First, I restate the conjecture here:

Conjecture 8.4.1: *Given* $U_{R,m,\delta} = \{\mathbf{IVP}_{\epsilon,C(x),t_s}, \mathbf{Rslt}_{t_f}\}$ *as in Definition 8.2.2 with* $t_f = 0$ *(note that* $t_f \in C(x)$*), any algorithm that correctly solves a subset of* $U_{R,m,\delta}$ *whose volume is non-negligible in* R^{2m} *(the total probability volume of the*

family $U_{R,m,\delta}$) will be a numerical integrator conforming to Definition 8.3.3.

Recall that Definition 8.3.3 simply states that the numerical integration routine takes the form $(\mathsf{Exp}^{\mathsf{A}}{}_k, \mathsf{NI}^{\mathsf{A}})$, a family of expansion algorithms, and then a stepping algorithm, respectively. This is a broad class of algorithms which affords single and multi-step integrators, and all known expansion algorithm families.

The conjecture limits a numeric integration algorithm to use only a finite amount of *local* information about the ODE during a given step. This chapter focuses on why it is acceptable to limit the expansion algorithms' knowledge of the global behavior of the IVP (in the case of IVPs as in Definition 8.2.2, derivatives of $r(t)$). I begin with revisiting the Kovacic algorithm.

Chapter 6 discusses the application of Differential Galois theory to the construction of the "Kovacic algorithm", a deterministic, symbolic algorithm that takes an ODE of the form $y''(t) = r(t)y(t)$ and returns either (a) the closed-form solution, or (b) that no closed form solution exists.

The first note is that the Kovacic algorithm returns *the space of closed-form solutions*, rather than a specific closed-form solution. (i.e., the algorithm returns $C_1 \cos(t) + C_2 \sin(t)$ with undefined constants C_1, C_2 when run on the ODE $y''(t) = -y(t)$, rather than a specific choice of C_1, C_2). Therefore, the application of the Kovacic algorithm to an IVP is overkill. Further, an IVP only requires a single solution for a given set of initial conditions, while the Kovacic algorithm returns the entire solution space.

This exemplifies the difference in approach between the Kovacic algorithm and more "standard" numerical integration techniques, such as Euler and Runge-Kutta methods. The Kovacic algorithm by definition requires *global* knowledge of the ODE. In the case where the IVP is of the form in Definition 8.2.2, the Kovacic algorithm must have an complete encoding for $r(t)$. On the other hand, the Kovacic algorithm is agnostic to the initial conditions.

Therefore, clearly the Kovacic algorithm is an example of an algorithm that would not be considered under Conjecture 8.4.1.

However, the Kovacic algorithm only returns useful information *if there exists a*

Liouvillian solution to the ODE. Otherwise, it returns nothing. It was not explicitly shown in Chapter 8 that most IVPs of the form in Definition 8.2.2 do not have closed form solutions, but it is indeed the case that virtually all of these IVPs do not have a closed form solution.

However, because the Kovacic algorithm exists for this class of ODEs, a guarantee can be provided that no such exact, closed-form solution exists. Recall that this is why ODEs of the form in Definition 8.1.4 were chosen. Conjecture 8.4.1 is only relevant if there does not exist a closed form solution (Otherwise, the Kovacic algorithm has already solved the ODE in all space).

Now, since there is no closed-form solution, Conjecture 8.4.1 is relevant, because all known solution techniques fall under the category of "expansions." In general, functions without closed form are not directly computable by discrete computers. Indeed, discrete computers are in general only capable of arithmetic operations (add, subtract, multiply, divide). Therefore, even simple functions such a square root, exponent, and sine are represented as interpolating polynomials (which are themselves expansions) [165].

There are two categories of expansions: Uniform and asymptotic.

9.1 Uniform Approximations

Consider an IVP of the form in Definition 8.1.4. The corresponding ordinary differential equation is by the definition of the form $y''(t) = r(t)y(t)$.

In this case, there are two functions that can be uniformly approximated: $r(t)$ and $y(t)$.

Although this statement cannot be proven, I conjecture that it is hard to provide a uniform approximation of $r(t)$ that results in a closed form solution. This is due to the conceptual reason that the uniform approximation results in piecewise or high-degree polynomial or exponential solutions. The solutions are analytically more complicated and do not in general make the system more likely to have a closed form solution. Moreover, to the author's knowledge, there do not exist any ODE solving algorithms

or methodologies that consider this approach.

Second, consider the uniform approximation of $y(t)$. Such methods include the Galerkin method and Finite Element Methods for boundary value problems [28]. Another class of uniform approximation methods includes regression, function expansion, and interpolation [165].

The primary common thread among each of these approximation methods is that global knowledge of the function is required as a pre-requisite to each method (e.g., data distributed throughout the area of interest for interpolation and regression, or data in certain limiting regions for boundary value methods). For the purposes of the security of the system, this is an important conceptual recognition.

First, initial value problems by definition are local in behavior, as the complete system state is given at a single, localized point in time, and the differential equation describes small-scale changes. This is largely incompatible with the above methods.

However, the attacker in the PPUF security game (Definition 8.5.6) can precompute on the class of differential equations. If the adversary can obtain some information regarding the global behavior of the system regardless of the challenge provided to the PAF, this would constitute a break in security. This is the reason for the selection of random elements from $U_{R,m,\delta}$ (cf. Definition 8.2.2). When selecting randomly from this family, an adversary cannot pre-compute the solutions (or a large enough subset thereof to interpolate), simply because the problem space is too large.

Therefore, I conjecture that uniform approximation methods are not advantageous for use in the context of a PPUF adversary.

9.2 Asymptotic Approximations

The second class of approximation algorithms are "asymptotic". Instead of looking for a closed-form solution that is correct over the entire real axis or complex plane, one looks for a closed-form solution that is valid over some subset defined where a certain parameter is large/small with respect to another parameter. This leads to the notion of an "expansion order."

Conjecture 8.4.1 states informally that any numerical integration scheme must be composed of a family of asymptotic expansions (by definition localized around some point or limit) indexed by their order (denoted $\mathsf{Exp}^{\mathsf{A}}{}_k$, and a stepping algorithm NI^{A}.

The classic example of such an asymptotic approximation method is the Taylor series expansion. In the case of an initial value problem, the function and its derivative are both known at the starting point at time t_s. Using the Frobenius method, additional derivatives can be computed. By taking $t \approx t_s$, the Taylor series expansion can be valid.

However, more complex asymptotic expansions (e.g., WKB, discussed later in this chapter and the Kovacic expansion, constructed later in this chapter) have significantly larger regions of accurate approximation when compared to a Taylor series expansion at equivalent expansion order. Therefore, when constructing a PPUF system or comparing the circuit complexity of analog and digital computing, it is important to bound the best possible performance of any asymptotic expansion algorithm in the context of an initial value problem.

To accomplish this, first consider *why* certain expansions are qualitatively "better" than others (e.g., Padé expansion consistently outperforms Taylor series expansion), even at the same expansion order.

Consider an order k Taylor series expansion,

$$\hat{y}_{\text{Taylor}}(t) = \sum_{i=0}^{k} C_i t^i$$

and an order (p, q) Padé expansion, where $p + q = k$:

$$\hat{y}_{\text{Pade}}(t) = \frac{\sum_{i=0}^{p} C_i^{\text{num}} t^i}{\sum_{j=0}^{q} C_j^{\text{den}} t^j}$$

Both expansions have the property that if the exact function is $y(t)$, then $y(t) - \hat{y}(t) = O(t^{k+1})$. However, the Taylor series expansion has the property that all higher derivatives of $\hat{y}_{\text{Taylor}}(t)$ are 0. This is not the case for the Padé expansion – the higher

order derivatives of $\hat{y}_{\text{Pade}}(t)$ are in general non-zero.

The behavior of these higher-order terms is the focus of Empirical Observation 8.4.2, and provides the foundation for the reduction of PPUF security and analog speedup to Conjecture 8.4.1.

9.3 Estimation of Sub-Dominant Expansion Terms

Conjecture 8.4.1 restricts the knowledge of $\mathsf{Exp}^{\mathsf{A}}{}_k$ of $r(t)$ to derivatives up to some maximal expansion order $k_{\max}$ at some time t_s. Therefore, because the IVP as in Definition 8.1.4 is of the form $y''(t) = r(t)y(t)$, $\mathsf{Exp}^{\mathsf{A}}(t)$ can only use the Frobenius method to compute derivatives of $y(t)$ at t_s up to order $k_{\max} + 2$ (the plus 2 term comes from the fact that the IVP consists of a second order ODE).[1]

However, $\mathsf{Exp}^{\mathsf{A}}{}_k$ is not completely ignorant of the higher orders of $y(t)$. This results from the fact that $\mathsf{Exp}^{\mathsf{A}}{}_k$ has implicit knowledge of the function space from which $r(t)$ is being drawn, and therefore also has knowledge of the function space from which $y(t)$ is being drawn.

Consider a toy example where $r(t)$ has the property that all of its derivatives have magnitude bounded to be less than some constant $D_{\max}$. $\mathsf{Exp}^{\mathsf{A}}{}_k$ therefore knows that $|y^{(k_{\max}+3)}(t_s)| < D_{\max}|y^{(k_{\max}+1)}|$. This represents a substantial restriction on the possible values of $y^{(k_{\max}+3)}$.

This provides some qualitative intuition why Padé expansion outperforms Taylor series expansion. In essence, Taylor series expansion "gives up" after expansion order $k_{\max}$, setting higher order derivatives of the expansion to 0. Padé expansion estimates these higher-order terms based on the assumption that the function being expanded is a rational polynomial.

These higher order terms are typically referred to as "sub-dominant." A higher-performing expansion algorithm will be able to estimate the sub-dominant terms well.

[1] Restriction of the knowledge of $r(t)$ appears to the author to be the most elegant way of restricting the capabilities of Exp. If Exp has global knowledge of $r(t)$, there is no theoretical reason why Exp itself can't be a numerical integrator that solves for $y(t)$ everywhere. It might be interesting to consider alternative formalizations of numerical integration that restrict Exp in a different way.

To study this, Chapter 8 considers higher-order coefficients of $r(t)$ as random variables, and Exp may attempt to estimate these random variables based on its knowledge of the structure of the space of ODEs. Informally speaking, the function space implies a probabliity distribution for the subdominant terms. The Exp algorithm in Chapter 8 computes this distribution and optimally chooses these subdominant terms.

In Chapter 8, Empirical Observation 8.4.2 formalizes the following:

Let $r(t)$ be drawn randomly from $U_{\mathrm{R,m},\delta}$ (cf. Definition 8.2.2). Even if an expansion algorithm knows (a) the first $k_{\max}$ derivatives of $r(t_s)$ and (b) complete knowledge of the function space (in this case $r(t) \in U_{\mathrm{R,m},\delta}$), there is still some randomness left over in the sub-dominant terms.

Furthermore, Empirical Observation 8.4.2 formalizes that not only is there entropy in the sub-dominant terms, there is also randomness in the output function $y(t_f)$ evaluated sufficiently far away from t_s. This randomness is characterized in terms of differential entropy.

This randomness limits the best possible performance of any asymptotic expansion algorithm, regardless of methodology used. However, while this analysis is useful from a theoretical perspective to prove the reduction in Chapter 8, the expansion algorithm presented in this chapter demonstrates concretely the efficacy of estimating sub-dominant terms based on knowledge of the function space by leveraging the Kovacic algorithm. It is referred to as the "Kovacic Expansion" in this thesis.

This algorithm, while not practical for everyday numerical integration usage, is useful as an instructive tool in understanding (a) approaches and limitations in estimating sub-dominant terms and (b) providing a real-world adversary for the results in Chapter 8.

9.4 Kovacic Expansion Methodology

Section 6.3 shows how the Kovacic algorithm provides a one-to-one mapping between fractional polynomials and Kovacic closed-form functions (Definition 6.3.1).

As a result, one may use Padé expansion in conjunction with this mapping to

construct an expansion (the "Kovacic Expansion" in Definition 6.3.3) of any order with a much larger function space: the set of Kovacic closed-form functions. This expansion is the subject of this section, and the high-level flow of this expansion algorithm is shown in Figure 9-1.

Recall that the purpose of constructing this expansion is to bound the step size of any numerical integration algorithm that is simulating an analog system. Recognize that this step size bound is imposed based on the error tolerance of the overall computation – the more error that can be tolerated, the larger the step size can be.[2]

This question may be stated precisely:

Given an ODE $y''(t) = r(t)y(t)$, initial conditions $y(t_s)$, and $y'(t_s)$, and an overall accuracy objective ϵ. What is the minimal number of expansions of order k that must be computed to approximate $y(t_f)$ to within ϵ as a function of t_f?[3]

Chapter 8 provides a lower bound on the number of steps required based on purely information theoretic grounds. The Kovacic expansion is constructed in an effort to find a real-world numerical integration algorithm that gets as close to the information theoretic bound as possible.

Before considering the Kovacic expansion, first consider how error accumulates in any numerical integration algorithm. At each step, the expansion algorithm computes an approximation function that matches the actual function to some finite order. The difference between this local approximation and the actual function is the *local truncation error.* The local truncation error is then compounded in each step into the *global truncation error* that describes the total error at the end of the numerical integration procedure (See Chapter 4 of [47]). One needs to set the step size to be small enough such that the global truncation error less than or equal to the required error tolerance (ϵ in the above example).

Traditional expansion algorithms (such as Runge-Kutta algorithms) use Taylor's theorem as well as known bounds on the derivatives of the independent variable $y(t)$

[2]Note that this translates into the noise requirement on the analog system – the more noisy an analog system is, the easier it is to simulate.

[3]A lower bound on the number of expansions to traverse from t_s to t_f is equivalent to an upper bound on the step size.

to calculate an upper bound on local truncation error [33]. This is impossible to perform for the Kovacic expansion, since the Kovacic expansion is not restricted to polynomial functions.

However, since the objective of the Kovacic expansion method is to provide a bound on the step size, rather than providing a practical alternative to traditional expansion algorithms, one does not need to provide an alternate algorithm for computation of this error. Instead, simply define the error in terms of the relation to the exact solution (which for a traditional expansion algorithm would be unknown). Therefore, considering the formalism for a single-step expansion algorithm in Definition 8.3.1, the Kovacic expansion returns a function $\hat{y}(t)$ with the maximal $\mathbf{R}$ such that $|y(t) - \hat{y}(t)| < \epsilon$ (the ϵ-accurate region).

Moreover, pessimistically define the local truncation error to be as high as the global truncation error. In a real system, this is never the case (global truncation error is always significantly larger than local truncation error). However, the global truncation error serves as an upper bound for the allowable local truncation error, which in turn allows for the calculation of an upper bound for step size.

Therefore, one can state that the maximal step size is bounded to be within $\mathbf{R} -$ since ϵ is the global truncation error, and the local truncation error must be less than or equal to the global truncation error.

9.4.1 Algorithmic Description

The Kovacic expansion is a novel expansion methodology that is given as arguments a linear second order differential equation with coefficients in $\mathbb{C}(t)$ (i.e., the Kovacic algorithm is applicable), a maximum expansion order k, a point of expansion t_s (that can be infinite), initial conditions $y(t_s)$, $y'(t_s)$,[4] and maximum local truncation error ϵ. The algorithm then returns the optimal Kovacic closed-form solution matching the Frobenius series of $y(t_s)$ to the maximum number of terms and having the largest step size within the ϵ-accurate region.

[4]The initial conditions don't have to be the same as the expansion point, but I choose this for convenience and because it reflects usage for numerical integration.

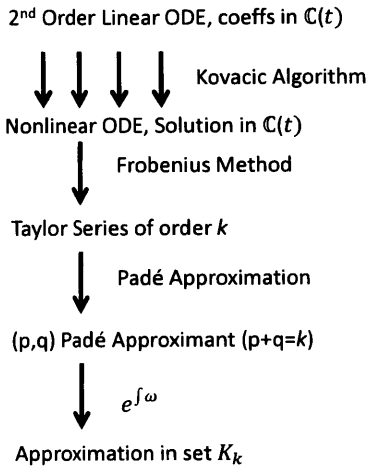


Figure 9-1: A depiction of the general flow of the order k Kovacic expansion. The ODE is processed by the Kovacic algorithm to yield a set of non-linear ODEs. Then, the Frobenius method [3] is used to obtain a degree-k Taylor series. One then computes all (p, q) Padé expansions of this series for $p+q = k$ and chooses the one with the largest step size within the ϵ-accurate region. One then returns the $y(t)$ that corresponds to this rational expression.

The high level algorithm can be summarized as follows, and the high-level flow is depicted in Figure 9-1. Simply put, this algorithm performs Padé approximation of the $a_{n-1}(t)$ term for each n and picks the one with the largest step size in $\mathbf{R}$.

$\{\hat{y}(t), \mathbf{R}\} = \mathsf{KovacicExpand}_{k,\epsilon}(\mathsf{ODE}(y(t), t), \{y(t_s), y'(t_s)\}, t_s)$:

1. For $n \in \{1, 2, 4, 6, 12\}$
   (a) Using $\mathsf{ODE}(y(t), t)$, compute the Kovacic ODE from Proposition 6.3.2. Let $a_{n-1}(t)$ denote the solution of this ODE.
   (b) Compute the expansion of $a_{n-1}(t)$ at $t \in \Gamma$, where Γ is the set of all poles of the coefficients in ODE and ∞.
   (c) Use a traditional ODE integrator to approximate the value of $a_{n-1}(t)$ at the requested t_s.
   (d) Compute the order $k + n - 2$ Frobenius series expansion at t_s [3].
   (e) Use (d) to compute all (p, q) Padé expansions of $a_{n-1}(t)$ at $t = t_s$ where $p + q = k + n - 2$ as $\{\hat{a}_{n-1,(p,q)}\}$.
   (f) For each Padé expansion $\hat{a}_{n-1} \in \{\hat{a}_{n-1,(p,q)}\}$

i. Compute a corresponding basis[5] $\{\hat{y}_1(t), \hat{y}_2(t)\}$ using Definition 6.3.1.

ii. Using $\{y(t_s), y'(t_s)\}$, use the basis from (1.f.i) to compute $\hat{y}(t)$ and add to GlobalList.

2. Use a traditional ODE integrator to approximate $y(t)$ in the neighborhood of t_s.
3. Using (2), find the $\hat{y}(t)$ in GlobalList with the largest step size in the ϵ-accurate region **R**.

9.4.2 Details of the Algorithm

Clearly, the algorithm description in Section 9.4.1 is much more complicated than a normal expansion algorithm, involving calls to other traditional numerical integrators in steps (1.c) and (2), as well as calculating multiple expansions and picking the one with the largest step size within the ϵ-accurate region.

This complication results from the relationship between $y(t)$, and $a_{n-1}(t)$ in three ways discussed below:

1. Derivation of $\hat{y}(t)$ from $\hat{a}_{n-1}(t)$: (steps 1.f.i-1.f.ii).
2. The approximation of $a_{n-1}(t_s)$: (steps 1.b, 1.c).
3. Comparing of the step sizes within the ϵ-accurate region: (steps 2, 3).

Deriving $\hat{y}(t)$ from $\hat{a}_{n-1}(t)$

The first complication derives from the basic ODE theory – a homogeneous second order ODE (like the one being studied – $\mathsf{ODE}(y(t), t)$) has a solution space that is two-dimensional (i.e., two free parameters). The initial conditions $\{y(t), y'(t)\}$ determine which element of this space is the path along which the independent variable $y(t)$ evolves. However, recalling Theorem 6.2.1 and Definition 6.3.1, the function $a_{n-1}(t)$ is *independent* of the initial conditions: it can only be used to compute the two-dimensional solution space as a whole.

[5] Given $\hat{a}_{n-1}$, one can compute a single element of this basis $\hat{y}_1(t)$. Using $\hat{y}_1(t)$, standard techniques can be used to find a second linearly independent basis function $\hat{y}_2(t)$ [151]. These two basis functions span the set of functions that are homogeneous solutions to $\mathsf{ODE}(y(t), t)$.

Therefore, in step 1.f.i, $\hat{a}_{n-1}$ must be used to compute a *basis* of the solution space. After this basis is computed, one must incorporate the initial conditions to solve for the actual path (step 1.f.ii).

Approximating $a_{n-1}(t_s)$

The second complication arises for similar reasons. The Kovacic ODE (computed in step 1.a) is inhomogeneous of order n. Therefore, it has in general a solution space consisting of the inhomogeneous solution added to the homogeneous solution. The homogeneous solution has some number of free parameters defined by "initial conditions" (unrelated to the two free parameters of the original ODE). The homogeneous solution space is a vector space of order n, where n is the number of free parameters.

The interpretation of the meaning of these parameters is much more subtle. Recall Proposition 6.3.2: if a closed-form solution exists to the original ODE, then *there exists a solution* $a_{n-1}(t) \in \mathbb{C}(t)$ to the secondary ODE. It does *not* state that the entire solution space for $a_{n-1}(t)$ is contained in $\mathbb{C}(t)$.

Put differently, the inhomogeneous solution to the Kovacic ODE is in $\mathbb{C}(t)$. However, the vector space of homogeneous solutions only is not necessarily entirely in $\mathbb{C}(t)$. Therefore, the inhomogeneous solution is the solution of interest, and the homogeneous solutions are not useful.

Now, reconsider the original problem – using Padé expansion to approximate $a_{n-1}(t)$ in the neighborhood of t_s. Like any expansion method, Padé expansion approximates a *single* solution (not the entire solution space). Therefore, one must supply "initial conditions" – the first n derivatives of $a_{n-1}(t)$ at t_s – in order to compute the Padé expansion.

However, as discussed above, one must pick initial conditions matching the value of the inhomogeneous solution *only*, since the homogeneous solutions are not desirable. This is, unfortunately, a non-trivial task.

However, there do exist methods to compute rational solutions to ODEs (see Chapter 4 of [149]). By calculating the indicial polynomial of the Kovacic ODE at any of the singularities of $r(t)$ (including $t = \infty$), one can construct a Laurent

expansion of the inhomogeneous solution around the singularity.

This method can be used to approximate *only* the inhomogeneous solution – separate from the homogeneous solution space. Therefore, one may approximate a point close to the singularity using the Laurent expansion, and then use a normal numerical integration technique to approximate $a_{n-1}(t_s)$. This value of $a_{n-1}(t_s)$ provides the appropriate initial conditions corresponding to *only* the inhomogeneous solution.

One now sees that the algorithm in Section 9.4 performs precisely these steps in steps 1.b and 1.c.

Picking $\hat{y}(t)$ with the Largest ϵ-accurate region

Recall that the objective of this expansion is to compute the expansion with the largest ϵ-accurate area. In order to formalize this objective, Definition 6.3.4 provides a set of functions for comparison, and the step size on the path of integration provides the optimality metric by which the "best" expansion is chosen (the ϵ-accurate region that affords the best step size is best).

This approach is reflected by the algorithm description in Section 9.4. Note that the algorithm in essence calculates the set of Kovacic expansions for $y(t)$ in step 1, while steps 2 and 3 pick the best expansion according the length of the step size.

Note, however, that in order to compare expansions, the algorithm simply uses a traditional ODE integrator to approximate $y(t)$ in the vicinity of t_s and then uses these data to compute and compare step sizes within the ϵ-accurate region for each expansion $\hat{y}(t)$.

Clearly this is very inefficient. However, it is a necessary step, as the comparison of step sizes within the ϵ-accurate region does not lend itself to a more analytic approach.

To illuminate this design choice, a short digression on the accuracy of traditional ODE integration techniques is useful. For most numerical methods (those that use Taylor series), one can bound the error of extrapolation by bounding the derivatives of the independent variable [33]. This bound decreases as the expansion increases order, corresponding to increased accuracy for a wider area.

However, this bound is an *upper bound.* For certain functions and error targets, it may be the case that the lower-order expansion has a larger step size[6]. There are no known general methods for lower-bounding the error of an expansion at a given order, which is required for the comparison of Kovacic expansions above [47].

Moreover, the above upper bound is only in place due to the properties of polynomials. This structure breaks down when considering the much more complicated space of Kovacic closed-form functions (Definition 6.3.1).

To end the digression, the existing mathematical machinery for comparison of function expansions is insufficient for the purposes of choosing the Kovacic expansion with the largest step size inside the ϵ-accurate region. Therefore, one must take a "brute-force" approach, manually searching through eligible functions using a precomputed approximation of the actual function.

Once again, this is highly inefficient, but the purpose of this algorithm is purely to theoretically justify an upper bound on the largest step size within the ϵ-accurate region, so such inefficiency is acceptable for these purposes.

Algorithm Overview

Now that the details of the Kovacic expansion algorithm have been described, the overall flow can be easily enumerated:

Step 1 computes all eligible Kovacic expansions of order k for the function $y(t)$. The computation of each expansion is broken into multiple steps, pictorally represented in Figure 9-1.

Steps 2 and 3 compare the set of Kovacic expansions of order k against an approximation of the actual solution, and pick the expansion with the largest step size within the ϵ-accurate region.

One sees that this construction of the Kovacic expansion is consistent with Section 9.4.1.

[6] e.g., consider the case where ϵ is large. Although higher order expansions are more accurate locally, they diverge faster. For large enough ϵ, a first-order expansion will have a larger step size within the ϵ-accurate region than many higher-order expansions.

9.5 Optimality of the Kovacic Expansion Approach

First, I begin with a Lemma regarding the construction of the Kovacic expansion as stated in Section 9.4. Note that this Lemma explains the additional $n-2$ term in Definition 6.3.3.

Lemma 9.5.1. *The order-k Kovacic expansion of $y(t)$ at t_0 returns $\hat{y}(t)$ such that $|y(t) - \hat{y}(t)| = O\left((t-t_0)^{k+1}\right)$ and the size of the step along the curve of integration (inside the ϵ-accurate region) is maximized.*

Proof. Recall that, for each given n, the Kovacic expansion algorithm computes a Frobenius series of a_{n-1} to order $k+n-2$. It then computes all possible (p,q) Padé expansions where $p+q = k+n-2$ using this Frobenius series. Therefore, ignoring defect in the Padé expansion (one can increase the order of the Frobenius series and Padé expansion to compensate), the expansion $\hat{a}_{n-1}$ has the property $|a_{n-1} - \hat{a}_{n-1}| = O\left((t-t_0)^{k+n-1}\right)$.

Recognize from Proposition 6.3.2 that $y = e^{\int \omega}$, where ω is defined in terms of an order-n polynomial including ω and up to its $n-1$'th derivative. Therefore, for a given n and expansion of a_{n-1} up to and including order $k+n-2$, this results in a match of ω up to and including order $k-1$.

Finally, computation of $e^{\int \omega}$ results in $|\hat{y}(t) - y(t)| = O\left((t-t_0)^{k+1}\right)$. □

Lemma 9.5.1 formally states the property achieved by the construction of the Kovacic expansion in Section 9.4.

Although this is a strong property, it is not the property needed to bound circuit complexity. Instead one needs a superlative condition like the one obtained through Conjecture 8.4.1 and Empirical Observation 8.4.2.

To provide evidence that the Kovacic expansion subsumes modern expansion algorithms, consider other expansion methodologies for second order linear ODEs: Taylor series expansion, Padé expansion, and WKB expansion.

9.5.1 Asymptotic Expansion Techniques - A Philosophical Perspective

In general, asymptotic expansions are defined in terms of a successive sum of continuous functions ϕ_n defined in terms of a limit point L. In particular, $\phi_{n+1}(t) = o(\phi_n(t))$ as $t \to L$. Depending on application, t may have different interpretations (e.g., a perturbation, some system parameter, an actual system variable such as time, or a system variable in some transform space, e.g., frequency). Depending on such applications different choices of ϕ_n are more appropriate [117].

This work is interested in *local expansions*, where t above is a system variable (time). Because the expansion is local, one can by definition restrict the arguments of the expansion to be only local information about the system – the system state at that particular time (in previous sections, this corresponds to the dependent variable $y(t)$ and its derivatives).

With this in mind, the simplest asymptotic expansion is the power series of order k. Such an expansion matches the first k derivatives of $y(t)$, and all sub-dominant terms (higher-order derivatives) are 0. However, this choice is not unique. One can easily consider asymptotic expansions that match the first k derivatives and have non-zero sub-dominant terms.

Therefore, the key question is how to choose the sub-dominant terms intelligently. The naïve answer of increasing expansion order by one is not useful, as the same question arises for the *next* subdominant term (and so forth). Instead, one should consider the *other knowledge* that one has about the function. In particular, if one knows the class that the original function belongs to (e.g., rational polynomials), then one can use an expansion in that function space to estimate the global properties of the function that result in the observed local behavior [165].

Informally, this concept of matching global structure based on local behavior is akin to function fitting, where measured data combined with a knowledge of the fit function result in the calculation of global parameters such as mean, standard deviation, etc. Much of the intuition behind function fitting is present in this informal

view of expansion:

- The size of the function space in the expansion is analogous to the number of parameters in the fit model.
- The order of expansion is analogous to the number of data points being fitted.
- The arithmetic precision of expansion is analogous to the precision of data being fitted.

Using this analogy, consider a fit model with a large number of parameters. This generally can increase the accuracy of the fit. However, if the number/precision of data points is insufficient, then overfitting can occur that reduces the accuracy of the model.

The intuition is identical for why an increased space of functions for an expansion algorithm generally results in a better expansion unless insufficient order or insufficient precision is used. Note that this observation is empirically confirmed with Padé expansions [165].

Although the above argument is purely informal, it provides evidence that an expansion algorithm with a larger function space (when supplied with correct numerical precision and expansion order), will perform better (have a larger step size inside the ϵ-accurate region) than an expansion algorithm that uses a smaller set of expansion functions (at equivalent order).

Note that this improvement is observed empirically with respect to Padé and Taylor expansions, and has a long history of usage in numerical analysis [165].

With this in mind, the next section will prove that the set of functions that are Kovacic expansions (Definition 6.3.4) subsumes the set of functions that are Taylor, Padé, and WKB expansions.

9.5.2 Kovacic Expansion Subsumes Existing Expansions

It is empirically observed that an expansion algorithm that searches a larger space of functions converges faster (has a larger ϵ-accurate area) than one with a smaller set

of functions. Therefore, it is shown in this section that the function space covered by the Kovacic expansion subsumes the function space of Taylor series expansion, Padé expansion, and WKB expansion.

Lemma 9.5.2. *The order-k Kovacic expansion subsumes the set of Taylor series expansions up to order* $(k+1)/2$.

Proof. Recognizing that for $n = 1$, $y = e^{\int \omega}$ for $\omega \in \mathbb{C}(t)$, it is known that one can represent $y(t) = p(t) \in \mathbb{C}(t)$ by setting $\omega(t) = p'(t)/p(t) \in \mathbb{C}(t)$. For a degree-$d$ polynomial, this would correspond to an order $(d-1, d)$ degree fractional polynomial for ω.

Setting the Kovacic expansion order to k, this would correspond to $d = (k+1)/2$. □

Lemma 9.5.3. *The order-k Kovacic expansion subsumes the set of* (p, q) *Padé expansions* $(p, q = k)$ *up to order* $(k+1)/2$.

Proof. Recognizing that for $n = 1$, $y = e^{\int \omega}$ for $\omega \in \mathbb{C}(t)$, it is known that one can write $y(t) = f(t) \in \mathbb{C}(t)$ by setting $\omega(t) = f'(t)/f(t) \in \mathbb{C}(t)$. Putting $f(t) = n(t)/d(t)$, where $n(t)$, $d(t)$ are degree p, q polynomials respectively, $f'/f = \frac{n'd-d'n}{d^2} \times \frac{d}{n} = \frac{n'd-d'n}{nd}$.

Let $p + q = r$. This corresponds to a $(r-1, r)$ degree fractional polynomial, so this would correspond to a $2r-1$-degree Kovacic expansion.

Therefore, an order-k Kovacic expansion can return a Padé expansion of order $(k+1)/2$. □

Lemma 9.5.4. *An order-*k_{kov} *Kovacic expansion subsumes the set of WKB expansions of order* $\frac{1-2k+k_{\text{kov}}}{6k}$, *where* $k = p + q$, *and* $r(t)$ *is a degree* (p, q) *fractional polynomial.*

Proof. For an $n = 2$ Kovacic expansion, recall the differential equation for a_{n-1}. Put $a_{n-1}(t)$ as $a(t)$.

$$a(t)^3 - 4a(t)r(t) + 3a(t)a'(t) - 2r'(t) + a''(t) = 0$$

The WKB expansion assumes that $r(t) = \lambda R(t)$, where $\lambda >> 1$. One can put:

$$a(t) = \sum_{i=0}^{k} a_i(t)\lambda^{-i}$$

Using this substitution, solve for each a_i in succession. Doing this yields the following form for $i > 0$ (where c_i is a constant):

$$a_i(t) = c_i \frac{r'(t)^{2i+1}}{r(t)^{3i+1}}$$

Note that since $r(t) \in \mathbb{C}(t)$, this implies that each $a_i(t) \in \mathbb{C}(t)$, which implies that $a(t) \in \mathbb{C}(t)$. Therefore, at this point, it is proven that the above WKB expansion is in the set of Kovacic closed-form functions.

Next, calculate the degree of the numerator and denominator of $a(t)$ to find the equivalent Kovacic expansion order. Let $r(t) = n(t)/d(t)$, where $n(t)$ and $d(t)$ are degree p, q polynomials, respectively. Making this substitution, $a_i(t)$ becomes:

$$a_i(t) = c_i \frac{(n'(t)d(t) - n(t)d'(t))^{1+2i}}{n(t)^{1+3i}d(t)^{1+i}}$$

This is degree $((p+q-1) \times (1+2i), (1+3i)p + (1+i)q)$.

When one combines all of the a_i terms, the combination of a_0 and a_i results in the highest-order numerator. Moreover, the degree is highest when $p = k$, $q = 0$ (letting $p + q = k$ be the total order of $r(t)$).

Plugging in these values, the numerator degree is $(3i+1)k-1$, and the denominator degree is $(1 + 3i)k$.

Therefore, the overall degree required of the Kovacic expansion (k_{kov}) is given by: $k_{\text{kov}} = 2k(1 + 3i) - 1$.

Where $k = p + q$, and $r(t)$ is a degree (p, q) fractional polynomial, and i is the order of the WKB expansion.

Therefore, given k_{kov}, the maximal WKB order that can be calculated is:

$$\frac{1 - 2k + k_{\text{kov}}}{6k}$$

□

The above lemmas prove that the function space of the Kovacic expansion subsumes a large class of modern expansions.

9.5.3 Conclusion

Based on these Lemmas and the empirically observed relationship between the size of function space and ϵ-accurate region at a given expansion order, I show that the Kovacic expansion outperforms other expansions, as it subsumes their function spaces.

There is the empirically observed relationship that an expansion algorithm family Exp_k with a larger function space will have a larger ϵ-accurate region than any expansion algorithm family Exp'_k whose function space is subsumed by the function space of Exp_k at equivalent expansion order (assuming adequate numerical arithmetic precision). This assertion is supported by empirical evidence with respect to the performance and application of the Padé and WKB expansions [165, 117, 6, 13].

The second observation is that the function space of the Kovacic expansion algorithm subsumes all modern asymptotic expansion methodologies for linear ordinary differential equations [170, 169, 33].

Before concluding, there are several subtleties regarding this expansion that should be noted. First, recognize that the objective is to maximize ϵ-accurate region when compared "at equivalent expansion order." This equivalence for different expansions is defined by Lemmas 9.5.2, 9.5.3, and 9.5.4. However, as observed in Section 9.5.1, the relationship between expansion order and the step size inside the ϵ-accurate region may not be purely monotonic depending on the ODE and desired ϵ (even though the size does asymptotically increase with order).

If one requires the maximum step size inside the ϵ-accurate region for all expansion orders $\leq k$, one can simply extend the Kovacic expansion algorithm to execute and then brute force search over all expansions of order less than k. In this case, it is trivially true that the step size within the ϵ-accurate region monotonically increases with maximum expansion order k.

The second subtlety that should be mentioned is how the Kovacic expansion handles ODEs with closed form solutions. Recall that if this is the case, then $a_{n-1}(t) \in \mathbb{C}(t)$. As a result, the Padé expansion will terminate at some point, yielding $a_{n-1}(t)$ exactly (assuming sufficient numerical arithmetic precision). This is because Padé expansions have been proven to converge to functions of this form over $\mathbb{C}$ except for potential spurious pole-zero pairs that result in negligibly small areas of inaccuracy (the Padé expansion converges "in measure") [13, 153].

Finally, there have been several points in this section that have referred to the requirement for "sufficient numerical arithmetic precision." This requirement for numerical precision is also discussed in Section 9.5.1 (it is analogized with function fitting). In essence, the working precision is chosen such that, given an output precision $\pm\delta$ for each expansion coefficient, the difference between the result computed with "sufficient" working precision and the result computed with infinite precision is less than δ. This prevents numerical breakdown in the Padé expansion as discussed in Section 9.5.1.

10 - Implementation Considerations and Future Work

To this point in the thesis, the work on Public Model Physical Unclonable Functions has been purely theoretical. This is in large part due to the complex theoretical requirement of a computational speedup of the PPUF hardware over a CMOS model. To that end, a formalism and theory is constructed in Chapter 8. Now that the theory and formalism have been proposed, it is both instructive and important to any future work to consider how such a PPUF device could be implemented in practice.

This is a significant physical challenge, requiring speed and control of an alternate computational modality. While CMOS is well-developed, there are few physical systems with sufficient speed and sufficient stimulation/readout technology to provide the constant factors required for PPUF operation.

This chapter has two parts. First, I discuss several physical systems that were considered, but rejected due to limitations of the physical system itself. These systems (classical optics and photonic crystals) are instructive in how they fail, as they conceptually map to failures in the three informal criteria discussed in Chapter 5. However, I identify a system (optical ring resonators) that shows promise in their use for PPUF systems.

Second, I dive a little deeper into the technology ecosystem surrounding optical ring resonators to provide architectural guidance for how a PPUF system might be constructed, where possible issues may arise, and what critical technological factors

will either enable or prevent the creation of a PPUF system using optical ring resonators as a platform.

10.1 Criterion 1 Failure: Optical Vector-Matrix Multiply

In the discussion of quantum linear optics, it was recognized that one needed to input a significant number of identical photons into a linear optical system. Remember that this requirement for identical photon generation presented the primary challenge in implementation, which rendered that approach infeasible for the foreseeable future. If one relaxes this requirement – not requiring the photons to be identical (i.e., using coherent states), then the linear optical system behaves purely according to classical mechanics. Such a classical linear system is similar to its quantum counterpart, but does not provide the same speedup.

Mathematically speaking, the system has n input ports, each of which has input light with a magnitude and phase. The system has m output ports, each of which will have a magnitude and phase. The input signal can be represented as a complex vector of dimension n: $\overline{X}$. The output vector is similarly represented $\overline{Y}$. The relationship can be stated simply: $\overline{Y} = M\overline{X}$, where M is a matrix describing the operation of optical system between input and output.

Although the matrix-multiply operation is much less complex than the computation of the permanent, one may still hope that such a system could still create a speedup. In particular, since photons are bosons, multiple beams of light can be "summed" for free by simply shining them at a single detector. In addition, since photons pass through each other, one can route photons from source to sink very easily (no wires are needed since two beams do not interact).

As a result, one can imagine a very efficient vector-matrix multiplication system that leverages these effects and could potentially provide significant speedup over CMOS adders/multipliers.

Tamir et. al. investigated the performance of an optical vector-matrix multiplier (VMM) and theoretically set the performance of their 256×256 matrix-multiplication module to 16 teraflops (optimistically) [159, 172]. The architecture of their system directly encodes the n input sources as n light sources and the m outputs as m detectors. In between these detectors is a macroscopic lens structure and a liquid crystal screen that is modulated to create arbitrary M.

This performance figure (16 teraflops) is on par or less than par with current GPU technology. In addition, the above VMM implementation was performing 8-bit integer arithmetic. A modern GPU computes using 64-bit arithmetic. A 64-bit double operation is at least 8-times more costly than an 8-bit operation. This results in an upper bound of 1-2 teraflops of GPU computation needed to exactly mimic the above system. This is comparable to 2 high-end consumer GPUs. Modern supercomputers can compute well into the petaflop range. Clearly, this system does not meet the requirements for providing a measurable speedup.

However, demonstrating a speedup over CMOS was not the primary goal of the above research. Therefore, it is worth considering this problem at a more abstract level to understand more generally the physical origin of the potential "speedup" and technological roadblocks for such an implementation.

When considering potential parameters with which to determine a speedup (constant factor or asymptotic), there are two parameters that one can use to scale the size of the matrix-multiplication problem. First, one can scale the size of the matrix and vectors themselves. Second, one can consider multiplying several constant-size matrices, and scaling the number of matrices to be multiplied.

10.1.1 Scaling Matrix Size

The first recognition when considering scaling the matrix size is that one no longer needs to encode arbitrary matrices. A static matrix can be considered. When taken as such, the classical linear optical system becomes very similar to the optical PUF presented by Pappu et. al. in 2002 [121, 146]. The architecture described by Pappu et. al. in 2002 has an input of multiple photons incident on a coherent scattering

medium. A quantum input/output version of this is presented by Goorden et. al. in 2013 [66]. From the latter version, it is clear that the input photons can be represented by a spatial mode, the scattering medium projects this mode onto another mode, and the final mode is measured.

The primary difference between the proposed matrix-multiply system and Pappu's proposal is that Pappu's proposal has a single moving light source and an array of detectors. A matrix-multiply system would have an array of static light sources and an array of detectors. With this in mind, consider the following PPUF hardware proposal:

Consider N input lasers directed into the colloidal object. N photodetectors are on the other side. If the amplitudes of the input lasers are given as A_n, and the amplitudes present at the photodetectors are given as D_n, then the overall operation can be described as $D_n = \sum_n I_{n'} M_{nn'}$, where $M_{nn'}$ is the connection matrix between different lasers and each photodetector. The optical system described above operates in the coherent scattering regime because the optical elements in the colloid are much larger than the wavelength. Therefore, the above A_n, $M_{nn'}$, and D_n are complex to represent phase. An actual detector measures intensity, not amplitude, which is $|D_n|^2$. However, this does not introduce any computational issues.

For this PPUF hardware, the challenge bits would be mapped to the input vector representing intensity/phase. The response bits would be derived from the intensity/phase (or just intensity) vector as measured by the detectors. The computation would be the calculation of the output vector. Consider the case where there is one input and one output (i.e., one large matrix multiply operation is performed).

Using this model, one can analyze the physical requirements as follows:

The first clear assumption is that the above scattering system has a model that accurately computes the behavior of the physical system. Fortunately, while it is difficult to generate a model by direct estimation of the sphere locations (via microscopy or other methods), this is not necessary. A key point here is that the *only* property of the scattering device that is being probed is its scattering pattern. In other words, the information contained by the locations of each of the spheres has

additional information not contained by just the speckle images (it is possible for two arrangements of spheres to have the same speckle pattern output). Therefore, it is in general more efficient to simply measure the output patterns created by an orthogonal basis of input modes. The output speckle patterns add linearly, so deriving a new speckle pattern is simply a projection operation (matrix multiplication).

With this problem interpretation, a "speedup" can potentially be derived between the PPUF hardware and the CMOS implementation. This speedup would originate from the fact that the "computation time" of the scattering medium is defined by the maximum optical path length divided by the speed of light. This path length depends in some complex way on the "size" of the computation. However, it turns out not to be necessary to consider these issues.

Although the matrix-multiply operation is computed quickly in an optical system, the mathematical abstraction results in extremely efficient implementations in CMOS, and unacceptable overhead for the optical system. To observe this, consider the scale of the system to be constructed to achieve a measurable speedup.

The term "measurable" implies that the timing difference should be discernible even when one includes a random network latency on the order of 100ms.

Therefore, a best-effort digital implementation of this single matrix multiplication must require a computation time on the order of 0.1-1 sec. With modern GPU architectures (ignoring supercomputers), the matrix size must be at least on the order of 10^6. This is in part due to the fact that the laser intensities are analog signals and can (optimistically) be measured to roughly 16 bits. Even a 24-bit ADC implies that the calculation can be simulated using integer arithmetic. Most GPUs and supercomputing architectures implement 64-bit arithmetic.

An input size of 10^6 implies that the optical implementation is infeasible for integrated applications. Integration into any silicon process means that an IR wavelength must be used (1500nm) because Si is not transparent at shorter wavelengths. As a result, this also limits the size of the laser sources, waveguides, and detectors. A waveguide must be at least as wide as the wavelength. If all input light sources are piped independently into the colloid (as they must be in order to represent a large

number of challenges), then the waveguides must be aligned next to each other. This implies that the surface area of the colloidal substance must be on the order $1m^2$. This is not feasible in an integrated architecture.

Therefore, it can be concluded that even if matrix multiplication were faster than CMOS, a one-shot matrix multiply operation is not a suitable platform from which to derive a significant separation between optoelectronics and CMOS processes due to the physical overhead of the input/output systems.

10.1.2 Conclusion

In conclusion, the above approach of classical linear optics is unlikely to work due to the reasons presented above for the PPUF application. In each case described above, a more abstract mathematical problem was encoded into the physical system, and in each case, this encoding was the origin of the failure of the proposed PPUF hardware device.

For the first option (scaling the matrix size), the hardware required to encode the problem into a physical system became unwieldy. For the second option (scaling the number of matrix operations), the encoding process did not also encode all of the structure of the mathematical problem into the hardware device, so a CMOS model can exploit this extra structure to out-compute the hardware device.

The lesson learned from this approach results in Criterion 1: one should not encode a more abstract mathematical problem into a physical system (cf. Section 5.6). The problem of simulating the physical system itself should be the "problem" to be solved by the PPUF model.

Another way of stating this criterion is that any mathematical structure to the "problem" being solved by the PPUF that should be inherently baked into the physical system itself.

10.2 Criterion 2 Failure: Geometry Modulation

The above system failed in part because once the matrix elements of the optical system had been characterized, the actual matrix-vector multiplication problem did not allow for sufficient speedup over an CMOS computer.

Therefore, consider changing the matrix elements themselves by modulating the geometry of the optical system. Consider the following proposed PPUF hardware system: n optical inputs to an element that is coupled to n optical outputs.

The challenge bits are used to encode a certain geometry of the optical element. The response bits are derived from the magnitude/phase of the output elements.

The fundamental difference is that the coupling matrix defining the input/output relationship of the PPUF hardware is no longer defined by a static optical system. The modulation of the geometry will have complex, nonlinear effects on the matrix elements of the system. Therefore, the speedup is no longer derived from the computational time to execute the linear operation of the coupling matrix, but from the computational time to derive the matrix elements.

A model wishing to calculate the correct response bits would need to calculate the magnitude and phase at the optical output. Such systems are solved in the real world by commercial PDE solvers such as Comsol, meep, and others [73]. These systems discretize the PDE by creating a mesh, and use the mesh to break the PDE into a large number of coupled ODEs.

Assuming that the PPUF model would use a method similar to the above, the question of speedup becomes more complicated and difficult to formalize. In general, there are two aspects to the model that can be computationally intensive: (1) the generation of the mesh and defining the coupled ODEs, and (2) solving the system of ODEs.

Depending on the modeling algorithm being used, task (1) may or may not be difficult. For example, many FDTD systems dedicate effectively zero computational time to the mesh and create a square mesh without any optimization, while FEM solvers spend significant time optimizing the mesh. Ultimately, the optimization of

the mesh goes only to obtaining a more accurate solution using fewer coupled ODEs.

Task (2), namely solving the system of coupled ODEs, may also be the most computationally intensive. Unfortunately, there are a number of methods that a model can use to circumvent the full simulation of the system or speed up the simulation dramatically using parallelism. These are outlined below.

First, there has been significant and recent study into the theory of model order reduction [143, 127, 27, 38, 144]. This field is dedicated to the study of taking a very high-order ODE (i.e., created by discretizing a PDE in this example) and approximating with a lower-order ODE system. Systems have been identified that are easily modeled using these methods. However, there has been much less progress in identifying lower bounds on the simulation complexity of a given system – especially in the case of nonlinear systems. Therefore, given a set of coupled ODEs, it is not clear in general how many ODEs actually need to be simulated to obtain a reasonably approximate solution for the original set.

Second, the simulation of coupled ODEs is an inherently parallelizable problem. Numerical simulation of such systems will convert the high-order ODE into a high-order recurrence relation and numerically step through the recurrence. In this recurrence relation, each node of the mesh is stepped individually. In addition, each node only requires information from the past step of itself and its adjacent nodes to compute the next iteration. This is highly parallelizable, as the computation and communication is localized. Although current processing systems have a significantly smaller number of cores than the number of nodes in a mesh, current processing systems have significantly more complex CPU and communication architectures not needed in this type of simulation. There does not appear to be a clear technical boundary preventing massively parallel simulation of such a system.

10.2.1 Conclusion

The proposed PPUF hardware fares significantly better than optical matrix multiplication. However, the speedup of such a system would be largely conjectural based on the following statements that don't seem to have a clear reduction to a well-studied

mathematical problem:

- The relationship between the geometry of the optical element and its stimulation response models a random oracle.
- The simulation of the optical element cannot be sufficiently parallelized to reduce simulation time.
- As the size of the optical element grows, the time complexity of simulating the optical element grows.

Unfortunately, the inherent parallelism of this problem and recent advances in model order reduction imply that such a scale factor does not have a clear lower bound. Therefore, it is not recommended that the speedup of the PPUF hardware be based on such an approach. This is identified as Criterion 2 in Section 5.7.

10.3 Ring Resonators - Building ODEs with Optics

In Chapter 5, I claim that the PPUF hardware define some ordinary differential equation that the PPUF model is required to simulate. It was observed earlier in this chapter how the PPUF system fails when this is not the case.

Consider technology that operates in the optical frequency range, as these technologies are physically small in scale, and have been studied extensively for applications in computation and communication. In this section, I propose the use of integrated microring resonators as a potential PPUF platform.

This section shows how to implement a second-order ordinary differential equation with very fast internal timescales using integrated microring resonators. I find that this technology could potentially be applicable for usage as a platform for PPUF technology. However, further investigation is needed to determine the constant factor relationship between the CMOS simulator and an microring resonator implementation of PPUF hardware.

10.3.1 Microring Resonator Physics

A ring resonator consists of two or more waveguides coupled through physical proximity. The canonical single ring resonator is shown in Figure 10-1. The light is confined to the waveguide except in the coupling regions, where energy transfers between the the two waveguides.

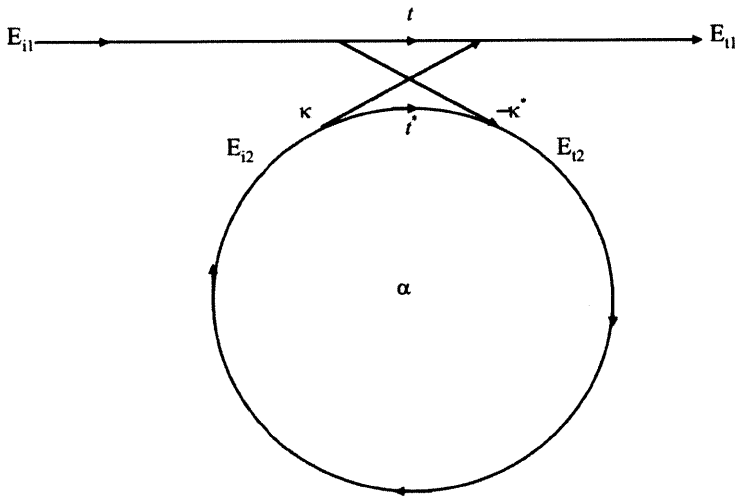


Figure 10-1: The canonical single ring resonator. Note that the ring is coupled to a waveguide on top of the ring [125].

Consider the coupling between the ring resonator in Figure 10-1 and the top waveguide (the "Input Port" and "Throughput Port"). The coupling is given by:

$$\begin{pmatrix} E_{t1} \\ E_{t2} \end{pmatrix} = \begin{pmatrix} t & \kappa \\ -\kappa^* & t^* \end{pmatrix} \begin{pmatrix} E_{i1} \\ E_{i2} \end{pmatrix}$$

Note that the coupling coefficients obey the equation $|\kappa^2| + |t^2| = 1$. Finally, the ring itself results in the relationship:

$$E_{i2} = \alpha E_{t2}$$

The α term is complex-valued with $|\alpha| \leq 1$. Its magnitude corresponds to the optical loss around the ring, and its phase corresponds to the phase delay around the ring.

Assume that the optical source turns on instantaneously, and the transient behavior at the throughput port is observed.

In this case, the concept of a "phase difference" around the ring is less interesting than the actual time delay around the ring. This time delay is the only time factor baked into the problem. Therefore, consider a discrete recurrence relation with step time Δt equal to this time delay around the ring. The ring resonator then has the following recurrence relation:

$$E_{i2}[n] = \alpha t^* E_{i2}[n-1] - \alpha\kappa E_{i1}[n-1]$$
$$E_{t1}[n] = \kappa E_{i2}[n] + t E_{i1}[n]$$

This corresponds to $E_{i2}[n]$ having an impulse response $(\alpha t^*)^n$. The resonance condition can be understood by studying the unit response. This response is (for $n > 0$):

$$E_{\text{step}}[n] = \sum_{k=1}^{n} (\alpha t^*)^{n-k} \alpha\kappa$$

Note that if αt^* is not real (corresponding to an out-of-phase excitation), the individual summands have varying phase. For real αt^*, the sum maximizes since all of the summands have the same phase (0 phase), corresponding to an on-resonance response.

Next, note that this recurrence relation is linear time-invariant. Therefore, one can analyze it using the Z-transform. This will not be a desirable property, as it would allow a model to be able to shortcut the computation of the time-evolution of the system. If the parameters of the ring resonator (e.g., α) are time-varying (a function of n), then the step response becomes:

$$E_{\text{step}}[n] = \sum_{k=1}^{n} \prod_{m=n-k}^{n} (\alpha[n] t^*)^m \alpha\kappa$$

In general, the above system does not have sufficiently complex dynamics to meet

the requirements needed. This can be seen intuitively since the recurrence relation is first order. As such, it is similar to a first-order differential equation.

Therefore, consider a dual ring-resonator system as shown in Figure 10-2. Not surprisingly, the double-ring structure will result in a second-order recurrence relationship. The objective will be to show that this recurrence relationship exhibits tunable oscillatory dynamics.

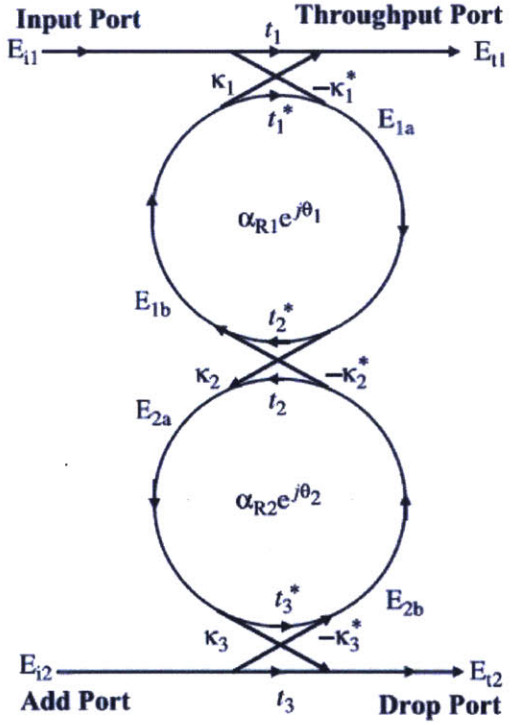


Figure 10-2: A series-connected dual ring resonator [125].

The mathematical relations between ports is given:

$$E_{1a} = -\kappa_1^* E_{i1} + t_1^* \alpha_1 e^{j\theta_1/2} E_{1b}$$

$$E_{1b} = t_2^* \alpha_1 e^{j\theta_1/2} E_{1a} - \kappa_2^* \alpha_2 e^{j\theta_2/2} E_{2b}$$

$$E_{2a} = \kappa_2 \alpha_1 e^{j\theta_2/2} E_{1a} + t_2 \alpha_2 e^{j\theta_2/2} E_{2b}$$

$$E_{2b} = -\kappa_3^* E_{i2} + t_3^* \alpha_2 e^{j\theta_2/2} E_{2a}$$

Note that in this case, the loss and phase accumulation around the ring resonator are broken out into α and θ, respectively. The recurrence relation is derived in the same manner as above:

$$E_{1a}[n] = t_1^* t_2^* \alpha_1 e^{j\theta_1} E_{1a}[n-1] - t_1^* \sqrt{\alpha_1} e^{j\theta_1/2} \kappa_2^* \sqrt{\alpha_2} e^{j\theta_2/2} E_{2b}[n] - \kappa^* E_{i1}[n]$$
$$E_{2b}[n] = t_3^* \sqrt{\alpha_2} e^{j\theta_2/2} \kappa_2 \sqrt{\alpha_1} e^{j\theta_1/2} E_{1a}[n-1] + t_3^* t_2 \alpha_2 e^{j\theta_2} E_{2b}[n-1]$$

Applying some example parameters results in following transient response shown in Figure 10-3.

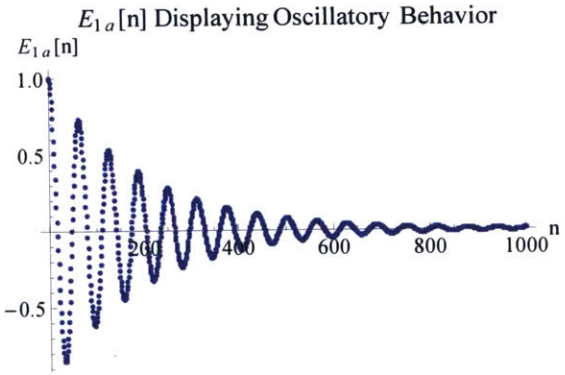


Figure 10-3: The E-field amplitude of the above dual ring-resonator displaying oscillatory behavior in its impulse response. The parameter being observed is E_{1a}.

Intuitively, what causes the oscillatory behavior is that the rings and input/output waveguides are weakly coupled. As a result, the initial impulse on E_{1a} results in a large amount of energy in ring 1, and zero energy in ring 2. The energy oscillates between the two rings, slowly leaking out through the output ports and each ring's internal losses. The oscillation frequency of this process is tunable depending on the coupling between the two rings and the input/output waveguides. The next step is to understand how to tune this process to achieve the correct oscillation frequency.

A significant simplifying assumption can be made for the dual-ring resonator. If the system is driven on resonance, then the phase accumulation around each loop ($\text{Arg}(t_1^* t_2^* e^{j\theta_1})$ for ring 1, and $\text{Arg}(t_2 t_3^* e^{j\theta_2})$ for ring 2) must be 2π. Without loss of generality, one can let $t_1, t_2, t_3 \in \mathbb{R}$ and $0 \leq t_1, t_2, t_3 \leq 1$. This can be seen as

follows. In effect, in setting $t_i \in \mathbb{R}$, any phase shift in the t parameter is baked into the θ parameter. While this is reasonable, one must show that the same can be done with the κ parameter. To that end, recognize that in this model, there is a single input and two output ports, all coupled via κ_1 and κ_3. Any phase introduced via these parameters can be recognized as an overall phase shift in the input and output. This is reflected in the fact that the recurrence equation for internal state does not depend on any of these variables.

The final value to consider is κ_2 and t_2. The key recognition here is that incorporating t_2's phase shift into the θ_i variables results in a constant overall phase shift in E_{2b} with respect to E_{1a}. Ultimately, if only the magnitude of the output signals are measured and relative phase is discarded, this relative phase shift is acceptable.

An interesting side note about the above assumption is that it only works in *linear chains* of ring resonators. If there is a "feedback path" that allows the phase accumulation between rings to feed back into the first ring, the assumption breaks down. This occurs in all $m \times n$ square lattices of ring resonators for $m > 1$ and $n > 1$.

With this in mind, the homogeneous solution of the above recurrence relation is:

$$\begin{aligned} E_{1a}[n] =& C_1 \left(\frac{1}{2}\right)^{n+1} \frac{(r-m)(p-r)^n + (r+m)(p+r)^n}{r} \\ &+ C_2 \left(\frac{1}{2}\right)^{n} \frac{((p-r)^n - (p+r)^n) t_1 \kappa_2 \sqrt{\alpha_1 \alpha_2}}{r} \end{aligned}$$

with the following parameter definitions:

$$\begin{aligned} p &= (t_1 + t_3) t_2 \alpha_2 \\ m &= (t_1 - t_3) t_2 \alpha_2 \\ r &= \sqrt{m^2 - 4 t_1 \alpha_1 t_3 \alpha_2 \kappa_2^2} \end{aligned}$$

Now, the objective is to observe the oscillation frequency of the recurrence relation given that each step represents an interval Δt corresponding to the propagation delay

of light around the ring resonator. This frequency can be observed by recognizing that, if t_1, t_2, t_3, $\in \mathbb{R}$ and $0 \leq t_1, t_2, t_3 \leq 1$, then the only term above that can be complex in the expression for $E_{1a}[n]$ is r. If this term is complex, then E_{1a} is seen to oscillate at $(\angle(p+r))/(2\pi) \times \Delta t$.

Therefore, the criteria for oscillation is to have r complex. Using the above expression for r and substituting $\kappa_2^2 = 1 - t_2^2$ results in the criterion:

$$t_2^2(t_1\alpha_1 + t_3\alpha_2)^2 - 4t_1\alpha_1 t_3\alpha_2 < 0$$

Observe that if oscillation is desired, then setting $t_1\alpha_1 = t_3\alpha_2 = \lambda$ reduces this criterion to $t_2^2 - 1 < 0$. This is always true for $0 \leq t_2 < 1$, which covers the entire range of t_2 (excluding $t_2 = 1$, which corresponds to completely uncoupled rings). The physical meaning of setting $t_1\alpha_2 = t_3\alpha_2$ is that the two rings are symmetric with respect to the optical loss around the ring.

Given that this oscillation is desired, one can re-analyze the original problem, making the two rings symmetric ($t_1 = t_3$, $\alpha_1 = \alpha_2$). The recurrence relation can then be written:

$$\begin{pmatrix} E_{1a}[n+1] \\ E_{2b}[n+1] \end{pmatrix} = t_1\alpha_1 \begin{pmatrix} t_2 & -\kappa_2 \\ \kappa_2 & t_2 \end{pmatrix} \begin{pmatrix} E_{1a} \\ E_{2b} \end{pmatrix} + \begin{pmatrix} -\kappa_1 \\ 0 \end{pmatrix} E_{i1}[n] \qquad (10.1)$$

If one defines $\cos(\theta) = t_2$, then the matrix in Equation 10.1 is easily seen as a rotation matrix by θ. Therefore, the impulse response is in general: $(t_1\alpha_1)^n \cos(\theta n)$.

Using this solution, one can translate the matrix recurrence relation in Equation 10.1 into a single second order recurrence. Note that Equation 10.2 is not true if the parameters α_1, t_1, or t_1 are functions of n.

$$E_{1a}[n+2] - 2t_2t_1\alpha_1 E_{1a}[n+1] + (t_1\alpha_1)^2 E_{1a}[n] = 0 \qquad (10.2)$$

This recurrence relation can be used with the theory presented in Chapter 8 by using time-scale calculus – recognizing that the interval Δt is small, and the

recurrence relation in Equation 10.2 can be seen as an approximate ODE.

Given a recurrence relation, one may use time-scale calculus to compute the differential equation approximated by the recurrence relation (assuming a certain Euler, Runge-Kutta, etc. method). This process is implemented by taking the derivative definition and interpreting it as a recurrence relation:

$$\frac{F[n+1] - F[n]}{\mathrm{dx}} = F'(x)$$

where $F[n] = F(n\mathrm{dx})$. Mapping this relation and a similar one for the second order differential onto the recurrence relation given in Equation 10.2 results in the following differential equation:

$$E''_{1a}(t) - 2(t_2 t_1 \alpha_1 - 1)E'_{1a}(t) + \left(1 + (t_1\alpha_1)^2 - 2(t_2 t_1 \alpha_1)\right) E_{1a}(t) = 0 \qquad (10.3)$$

Note that E_{1a} is now interpreted as a continuous variable in t. An analysis of the recurrence relation to the solution of this differential equation shows that they approximate each other. This is shown for an example set of parameters in Figure 10-4.

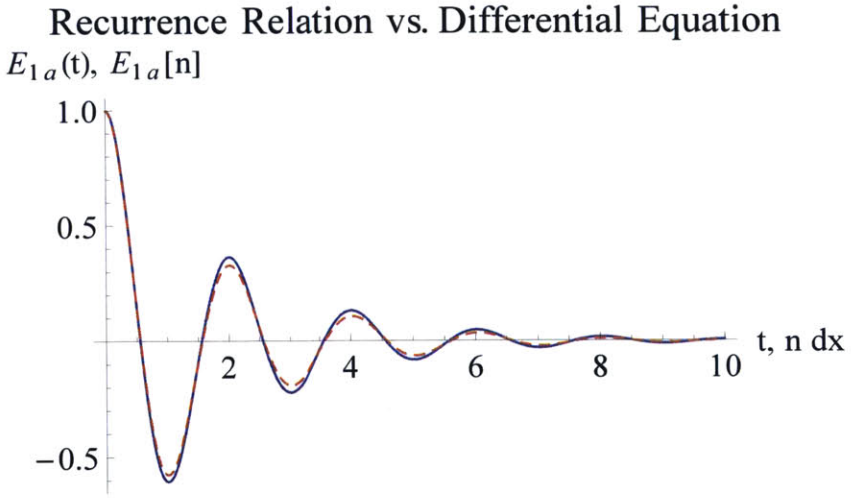


Figure 10-4: $E_{1a}[n]$ versus the derived differential equation $E_{1a}(t)$ on equivalent time scales. $E_{1a}[n]$ is blue, $E_{1a}(t)$ is red, dashed.

Note that there is a small error between the recurrence relation and the associated differential equation. This is the case because the act of directly substituting the derivative definition into the recurrence equation is exactly equivalent to a forward Euler approximation. In effect, the above figure shows a differential equation (the derived equation), and its forward Euler approximation (the original recurrence relation).

The usefulness of interpreting the recurrence equation as a differential equation will become apparent in the next section, as it helps ensure that an adversary cannot write down a closed-form expression for the value of E_{1a} for all time (as is possible above) and use this expression for simulation of the system, rather than a direct computation using the recurrence relation.

10.3.2 Constructing LTV ODEs using Ring Resonator Recurrence

All of the above analysis has been done for an LTI ring resonator system. However, in Chapter 8, I argued that the system will need to have more complex dynamics (LTV) in order to support viable PPUF operation. This can be achieved simply by modulating the system parameters (κ_i, t_i) in time according to the behavior defined in Chapter 8.

In conclusion, the ring resonator-based architecture represents a potentially viable architecture for PPUF hardware, as it is shown how to construct ODEs using ring resonators with very fast dynamics.

10.4 Directions for Future Work

Electro-optic systems using ring resonator technology are a possible approach to implement a PPUF system. By using time-scale calculus, one can implement an arbitrary LTV ODE by modulating ring resonator coupling. This can be used in conjunction with the PPUF theory in Chapter 8. However, there are many challenges to

the physical implementation of such a system. This section is dedicated to discussing some of these challenges.

10.4.1 Implementation Challenges

Based on a review of the state of the art in integrated photonic systems technology, I briefly point out several challenges that have been identified for the utilization of optoelectronic systems for a PPUF. These show that there is still a significant gap between the proposed theory, and a practical implementation of a PPUF. I recommend study in these directions, not only for the purposes of constructing a PPUF, but also because development of electro-optic components along these lines will aid in the construction of a more general analog computing platform, as shown in Chapter 8.

- Loss/Noise: Waveguides in integrated optoelectronic systems are lossy. This has two impacts on a PPUF system. First, it limits the maximum computation time before all energy is lost from the ring resonator. Second, before all of the energy is lost, it significantly decreases signal to noise ratio. This can be compensated for by including an in-situ optical amplifier. However, amplifiers also introduce noise, and will have to be carefully calibrated to prevent the introduction of unwanted signals via the response of the amplifier itself.
- Modulator contrast: In the theoretical model, ODE parameters were allowed to vary significantly. However, coupling parameters and modulation contrast are significantly limited by the geometry of the system, which materials are used, and fundamental technological limits. It is not clear to the author at this point how such limitations may impact the implementation of a PPUF system.
- Readout Speed: In the PPUF system described in Chapter 8, an iterated approach was taken where the system state affected the ODE at future times. This requires conversion of the optical signal to a digital signal (to use the random oracle, e.g., SHA1). The speed of this measurement and analog-to-digital conversion also impacts the speed of the overall system. As discussed in Section 8.5, this overhead must be taken into account when comparing the speed of the

PPUF with the digital simulator.

- Modulation Speed and Control: Finally, the result in Chapter 8 is predicated on whether the modulator speed is indeed fast enough to outpace any CMOS simulator. Further, Chapter 8 requires that the PPUF follow an ODE wherein the coefficient $r(t)$ is a fractional polynomial. This is not a typical signal that electro-optic modulators are used for. Therefore, future work must be performed in understanding the performance of electro-optic modulators in creating the required signals for the PPUF hardware.

10.4.2 Creating a PPUF Model

In addition to the implementation challenges discussed above, one must be able to accurately characterize the PPUF "model." In the case of ring resonators (cf. Section 10.3), this corresponds to the delay/attenuation parameter for each ring α, as well as the coupling parameters τ, κ between each ring.

Statically characterizing these parameters can be done through spectroscopy of the ring resonator [114]. However, these parameters must be modulated by some external signal (e.g., through an electro-optic modulator). Therefore during a PPUF challenge, each of these parameters are functions of time. To accurately model the PPUF, one must characterize the response of the modulator to the input stimulus [160].

The characterization of both the intrinsic parameters of the ring resonator as well as the transient behavior of the model comprise the PPUF "model." Knowledge of these parameters allow for the deterministic simulation of the PPUF in response to a given optical and analog signal.

Recall that unclonability arises from the inability to construct two PPUFs with a sufficiently similar set of parameters to produce the same challenge/response behavior. This unclonable behavior has been observed in optical ring resonator systems, as resonance frequencies of two identically designed rings being measurably different. As a result, many ring resonator applications are "tuned" thermally [114]. Although thermal tuning allows for more fine control of some parameters (primarily α), some

parameters (especially those relating to the modulator response) cannot be tuned.

10.4.3 Comparison to CMOS Dynamics

In order to argue that there exists a constant factor speedup, one now must compare the fundamental dynamics of the optoelectronic architecture to the computation time required by a CMOS model for a given step.

The second criterion identified for PPUF hardware is that the hardware must have internal dynamics that are significantly faster than the dynamics of the CMOS model. From a practical perspective, this corresponds to the following statement: A CMOS simulator computes the time evolution of the PPUF hardware in steps of Δt. The time required by the CMOS simulator to compute this step is greater than Δt.

For digital CMOS, the point of comparison would be propagation delay of gates shown in Figure 10-5. Note that although the process has improved dramatically over the past several years, the fundamental gate delay of simple CMOS elements is not changing nearly as dramatically, and likely will continue to stabilize.

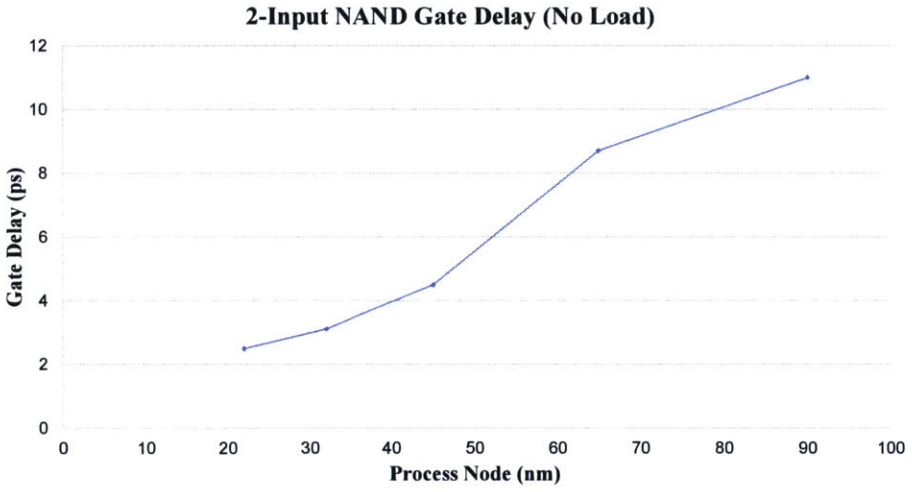


Figure 10-5: Plot of 2-input NAND gate delay versus process node [112, 118, 23].

With respect to the PPUF system, the absolute maximum step size of an analytic continuation algorithm will be based on the modulation speed. Therefore, the max-

imum step size of a simulation should be on the order of the minimum modulation time, identified above as 10ps.

Now, based on the recurrence relation describing the system, there will be at least one multiply operation to be performed during each step. If one requires only eight bits of precision, one can estimate the rough performance of a best-effort CMOS multiplier. Multiple different architectures have been studied in Intel 90nm technology [124]. The measured process performance shown in Figure 10-5 and predictive technology models in [150] result in an estimated improvement in gate delay of roughly $10\times$ when scaling to 14nm.

The 90nm study identifies a best implementation of a 8×4-bit multiplier as requiring at least 204ps of delay, corresponding to an optimistic estimate of 20ps of delay in 14nm.

Also, recognize that this comparison is optimistic in the favor of CMOS. First, an accurate optoelectronic simulation would likely require a larger 8×8-bit or 16×16-bit multiplication. Second, the recurrence relation will require more computation than a single multiplication operation to complete. Finally, this compares the *absolute maximum* step size of the system, allowing the computation of an arbitrary order Taylor series expansion. Real, optimized algorithms will be closer to an implementation of RK4, which will require a step size that is significantly smaller (10-100$\times$).

A rough layout of the various timescales of the system is given in Table 10.1. Note that $t_{\text{MODULATOR}}$ and t_{READOUT} represent the current state-of-the-art modulator and readout speeds. The ring resonator coupling time t_{OSC} is tunable over a wide range of possible timescales.

Table 10.1: Relative Timescales of Optoelectronic System Compared to CMOS

Decay time of ring resonator	$t_{\text{TRANSIENT}}$	$\sim$1 ns
CMOS Multiplier time	t_{CMOS}	$\sim$20 ps
Optoelectronic modulator timescale	$t_{\text{MODULATOR}}$	$\sim$ 5-10 ps
Optoelectronic readout timescale	t_{READOUT}	$\sim$ 5-10 ps
Characteristic time of ring resonator coupling	t_{OSC}	$\sim$ 5-10 ps
Optical delay around ring resonator	t_{RING}	$\sim$0.1 ps
Optical frequency	t_{OPTICAL}	$<<$0.1 ps

With these factors in mind, it is still too close to determine if optoelectronic systems will provide a usable platform for PPUF hardware and be faster than any CMOS model. However, no systemic problems have yet been identified, so further investigation is warranted.

11 - Conclusion

In this thesis, I have first resolved several outstanding issues with existing PUF systems. Learning Parity with Noise (LPN) is used to provide a computationally secure, practical fuzzy extractor construction that improves substantially over the state of the art. This construction is integrated into existing PUF formalism [10, 58], and a security reduction is provided from the definition of a fuzzy extractor to the hardness of LPN. Further, the theory of LPN is extended to understand what types of PUF distributions are allowed. The requirement on PUF bit distributions for the construction is stronger than a min-entropy requirement, but significantly weaker than an i.i.d. noise requirement, showing that if correlation can be estimated, the only change to the fuzzy extractor construction is in the selection of parameters.

I show how error profiles obtained from a Field Programmable Gate Array implementation of PUFs subject to wide environmental variation can be efficiently corrected using helper data sizes that are substantially smaller than the state of the art. As a result, the proposed LPN fuzzy extractor is more practical for use in real-world applications.

Second, I have presented a computationally secure construction of a stateless Physical Unclonable Function in this thesis based on precise hardness assumptions. This has been an open problem for over thirteen years since silicon PUFs were introduced in 2002 [64].

The construction is secure in the random oracle model under the difficulty of standard Learning Parity with Noise (LPN) and a variant LPN problem. The construction is noise-free; the responses during challenge-response generation and successful verification match exactly. This means that an entity with a single challenge-response pair

can authenticate the PUF any number of times by treating the response as a shared secret. All the protocols described in [63] are enabled by the construction.

This work also proposes for the first time a formalism describing Public Model Physical Unclonable Functions in terms of the relative speedup of one computational modality over another. Using this formalism, this work provides a definition of security and a formal PPUF construction (cf. Section 8.5).

These definitions represent a theoretical foundation on which the emerging technology of Public Model Physical Unclonable Functions may be studied. This foundation can be viewed as an extension of existing PUF formalism [10] (cf. Section 8.1).

Next, this work presents a mathematical conjecture (cf. Section 8.2) regarding the form of numerical integration algorithms. This work also presents significant justification of this conjecture in the context of previous work in the field of numerical integration. I present a reduction from the security guarantee for the PPUF construction to the conjecture (cf. Section 8.5). Finally, I provide recommendations towards the physical implementation of a PPUF system.

The theoretical foundation I present is necessary to enable further development in the field of Public Model Physical Unclonable Functions, and this work establishes that potential exists in the field of PPUF systems to enable "security without secrets."

Finally, this work recognizes that the key challenge for PPUF implementation is an instance of a broader complexity theoretic question regarding the relationship between analog and digital computing. In this work, I show a particular mathematical problem wherein analog computing does have a circuit complexity advantage over any digital simulator (assuming the conjecture mentioned above). This result provides the first constant-factor separation between analog and digital computing modalities.

Bibliography

[1] Mupad reference: Ode. `http://www.mathworks.com/help/symbolic/mupad_ref/ode.html`. Accessed: 2015-07-21.

[2] Some notes on internal implementation. `https://reference.wolfram.com/language/tutorial/SomeNotesOnInternalImplementation.html`. Accessed: 2015-07-21.

[3] Frobenius method. 2002. Accessed: 2015-07-21.

[4] Scott Aaronson. Guest column: Np-complete problems and physical reality. *ACM Sigact News*, 36(1):30–52, 2005.

[5] Scott Aaronson and Alex Arkhipov. The computational complexity of linear optics. In *Proceedings of the 43rd annual ACM symposium on Theory of computing*, STOC '11, pages 333–342, New York, NY, USA, 2011. ACM.

[6] Alexander Craig Aitken. On bernoulli's numerical solution of algebraic equations. *Proceedings of the Royal Society of Edinburgh*, 46:289–305, 1927.

[7] Adi Akavia, Shafi Goldwasser, and Vinod Vaikuntanathan. Simultaneous hardcore bits and cryptography against memory attacks. In *Theory of Cryptography*, pages 474–495. Springer, 2009.

[8] Benny Applebaum, Boaz Barak, and Avi Wigderson. Public-key cryptography from different assumptions. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pages 171–180. ACM, 2010.

[9] Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. Fast Cryptographic Primitives and Circular-Secure Encryption Based on Hard Learning Problems. In

Shai Halevi, editor, *Advances in Cryptology - CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 595–618. Springer Berlin Heidelberg, 2009.

[10] F. Armknecht, R. Maes, A. Sadeghi, O.-X. Standaert, and C. Wachsmann. A Formalization of the Security Features of Physical Functions. In *IEEE Symposium on Security and Privacy (S&P)*, pages 397–412, 2011.

[11] Sanjeev Arora and Rong Ge. New algorithms for learning in presence of errors. In *Automata, Languages and Programming*, pages 403–415. Springer, 2011.

[12] Andrew Baker. An introduction to galois theory. *University of Glasgow, lecture notes, retrieved from the address http://www. maths. gla. ac. uk/˜ ajb/dvi-ps/Galois. pdf.*

[13] GA Baker Jr and Peter Graves-Morris. Padé approximants. 1996. *Encyclopedia of Mathematics and its Applications*, 1996.

[14] G. T. Becker. The Gap Between Promise and Reality: On the Insecurity of XOR Arbiter PUFs. In *Cryptographic Hardware and Embedded Systems 2015 (CHES 2015)*, 2015.

[15] Georg T Becker, Alexander Wild, and Tim Güneysu. Security Analysis of Index-Based Syndrome Coding for PUF-Based Key Generation. In *IEEE International Symposium on Hardware Oriented Security and Trust, HOST 2015*, May 2015.

[16] N. Beckmann and M. Potkonjak. Hardware-based public-key cryptography with public physically unclonable functions. In *Information Hiding*, pages 206–220. Springer, 2009.

[17] Daniel J Bernstein and Tanja Lange. Never trust a bunny. In *Radio Frequency Identification. Security and Privacy Issues*, pages 137–148. Springer, 2013.

[18] Avrim Blum, Merrick Furst, Michael Kearns, and RichardJ. Lipton. Cryptographic Primitives Based on Hard Learning Problems. In DouglasR. Stinson, editor, *Advances in Cryptology - CRYPTO 93*, volume 773 of *Lecture Notes in Computer Science*, pages 278–291. Springer Berlin Heidelberg, 1994.

[19] Avrim Blum, Adam Kalai, and Hal Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. *Journal of the ACM (JACM)*, 50(4):506–519, 2003.

[20] Lenore Blum, Mike Shub, Steve Smale, et al. On a theory of computation and complexity over the real numbers: *np*-completeness, recursive functions and universal machines. *Bulletin (New Series) of the American Mathematical Society*, 21(1):1–46, 1989.

[21] Andrey Bogdanov, Miroslav Knežević, Gregor Leander, Deniz Toz, Kerem Varıcı, and Ingrid Verbauwhede. SPONGENT: A Lightweight Hash Function. In *Cryptographic Hardware and Embedded Systems - CHES 2011*, volume 6917 of *Lecture Notes in Computer Science*, pages 312–325. 2011.

[22] Sonia Bogos, Florian Tramer, and Serge Vaudenay. On solving lpn using bkw and variants. Technical report, Cryptology ePrint Archive, Report 2015/049, 2015.

[23] Mark Bohr and Kaizad Mistry. Intel's revolutionary 22 nm transistor technology. `http://download.intel.com/newsroom/kits/22nm/pdfs/22nm-Details_Presentation.pdf`.

[24] Olivier Bournez and Manuel L Campagnolo. A survey on continuous time computations. In *New Computational Paradigms*, pages 383–423. Springer, 2008.

[25] Olivier Bournez, Daniel S Graça, and Amaury Pouly. Continuous time models are equivalent to turing machines. *A characterization of P and NP with ordinary differential equations*, 2014.

[26] Xavier Boyen, Yevgeniy Dodis, Jonathan Katz, Rafail Ostrovsky, and Adam Smith. Secure Remote Authentication Using Biometric Data. In *EUROCRYPT'05*, pages 147–163, 2005.

[27] C Brennan, M Condon, and R Ivanov. Model order reduction of nonlinear dynamical systems. In *Progress in Industrial Mathematics at ECMI 2004*, pages 114–118. Springer, 2006.

[28] Susanne C Brenner and Ridgway Scott. *The mathematical theory of finite element methods*, volume 15. Springer Science & Business Media, 2008.

[29] Matthew A Broome, Alessandro Fedrizzi, Saleh Rahimi-Keshari, Justin Dove, Scott Aaronson, Timothy C Ralph, and Andrew G White. Photonic boson sampling in a tunable circuit. *Science*, 339(6121):794–798, 2013.

[30] Richard A Brualdi and Herbert J Ryser. *Combinatorial matrix theory*, volume 39. Cambridge University Press, 1991.

[31] D. Brunner, M. Soriano, and I. Fischer C. Mirasso. Parallel photonic information processing at gigabyte per second data rates using transient states. *Nature Communications*, 4(1364), 2013.

[32] Vannevar Bush. The differential analyzer. a new machine for solving differential equations. *Journal of the Franklin Institute*, 212(4):447–488, 1931.

[33] John C Butcher. Numerical methods for ordinary differential equations in the 20th century. *Journal of Computational and Applied Mathematics*, 125(1):1–29, 2000.

[34] John C Butcher. *Numerical methods for ordinary differential equations*. John Wiley & Sons, 2008.

[35] Phyllis J Cassidy, Richard C Churchill, Jerald J Kovacic, and William Sit. Introduction to differential galois theory. 2006.

[36] Q. Chen, G. Csaba, X. Ju, SB Natarajan, P. Lugli, M. Stutzmann, U. Schlichtmann, and U. Rührmair. Analog circuits for physical cryptography. In *International Symposium on Integrated Circuits (ISIC)*, pages 121–124, 2010.

[37] Q. Chen, G. Csaba, P. Lugli, U. Schlichtmann, M. Stutzmann, and U. Rührmair. Circuit-based approaches to SIMPL systems. *Journal of Circuits, Systems, and Computers*, 2010. to appear.

[38] Yong Chen. *Model order reduction for nonlinear systems*. PhD thesis, Citeseer, 1999.

[39] Marco Chiani and Davide Dardari. Improved exponential bounds and approximation for the q-function with application to average error probability computation. In *Global Telecommunications Conference, 2002. GLOBECOM'02. IEEE*, volume 2, pages 1399–1402. IEEE, 2002.

[40] Thomas M Cover and Joy A Thomas. *Elements of information theory.* John Wiley & Sons, 2012.

[41] Teresa Crespo, Zbigniew Hajto, and Juan José Morales Ruiz. *Introduction to differential Galois theory.* Wydawnictwo PK, 2007.

[42] G. Csaba, X. Ju, Z. Ma, Q. Chen, W. Porod, J. Schmidhuber, U. Schlichtmann, P. Lugli, and U. Ruhrmair. Application of mismatched cellular nonlinear networks for physical cryptography. In *International Workshop on Cellular Nanoscale Networks and Their Applications (CNNA)*, pages 1–6. IEEE, 2010.

[43] C. Dainty. *Laser Speckle and Related Phenomena.* Springer Verlag, 1984.

[44] Ivan Damgård and Sunoo Park. Is Public-Key Encryption Based on LPN Practical? *IACR Cryptology ePrint Archive*, 2012:699, 2012.

[45] Ramiz Daniel, Jacob R Rubens, Rahul Sarpeshkar, and Timothy K Lu. Synthetic analog computation in living cells. *Nature*, 497(7451):619–623, 2013.

[46] James H. Davenport and MF Singer. Elementary and liouvillian solutions of linear differential equations. *Journal of Symbolic Computation*, 2(3):237–260, 1986.

[47] Philip J Davis and Philip Rabinowitz. *Methods of numerical integration.* Courier Corporation, 2007.

[48] Jeroen Delvaux and Ingrid Verbauwhede. Side Channel Modeling Attacks on 65nm Arbiter PUFs Exploiting CMOS Device Noise. In *6th IEEE International Symposium on Hardware-Oriented Security and Trust - HOST 2013*, pages 137 – 142, 2013.

[49] Jeroen Delvaux and Ingrid Verbauwhede. Attacking PUF-Based Pattern Matching Key Generators via Helper Data Manipulation. In *Topics in Cryptology - CT-RSA 2014*, volume 8366 of *Lecture Notes in Computer Science*, pages 106–131. 2014.

[50] Vasil S Denchev, Sergio Boixo, Sergei V Isakov, Nan Ding, Ryan Babbush, Vadim Smelyanskiy, John Martinis, and Hartmut Neven. What is the computational value of finite range tunneling? *arXiv preprint arXiv:1512.02206*, 2015.

[51] S. Devadas. Non-networked RFID PUF authentication. US Patent Application 12/623,045, US Patent Number 8,683,210, 2008.

[52] S. Devadas, E. Suh, S. Paral, R. Sowell, T. Ziola, and V. Khandelwal. Design and implementation of PUF-Based "unclonable" RFID ICs for anti-counterfeiting and security applications. In *IEEE International Conference on RFID (RFID)*, pages 58–64, May 2008.

[53] Y. Dodis, B. Kanukurthi, J. Katz, L. Reyzin, and A. Smith. Robust Fuzzy Extractors and Authenticated Key Agreement From Close Secrets. *Information Theory, IEEE Transactions on*, 58(9):6207–6222, Sept 2012.

[54] Y. Dodis, L. Reyzin, and A. Smith. Fuzzy extractors: how to generate strong keys from biometrics and other noisy data. In *Advances in Cryptology - Eurocrypt 2004*, 2004.

[55] Anne Duval and Michèle Loday-Richaud. Kovacic's algorithm and its application to some families of special functions. *Applicable Algebra in Engineering, Communication and Computing*, 3(3):211–246, 1992.

[56] Alan Edelman and N Raj Rao. Random matrix theory. *Acta Numerica*, 14(1):233–297, 2005.

[57] Richard P Feynman. Simulating physics with computers. *International journal of theoretical physics*, 21(6):467–488, 1982.

[58] Benjamin Fuller, Xianrui Meng, and Leonid Reyzin. Computational fuzzy extractors. In *Advances in Cryptology-ASIACRYPT 2013*, pages 174–193. Springer, 2013.

[59] Fatemeh Ganji, Juliane Krämer, Jean-Pierre Seifert, and Shahin Tajik. Lattice basis reduction attack against physically unclonable functions. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 1070–1080. ACM, 2015.

[60] Mingze Gao, Khai Lai, and Gang Qu. A Highly Flexible Ring Oscillator PUF. In *The 51st Annual Design Automation Conference 2014, DAC '14*, pages 89:1–89:6, 2014.

[61] B. Gassend, D. Lim, D. Clarke, M. van Dijk, and S. Devadas. Identification and authentication of integrated circuits. *Concurrency and Computation: Practice and Experience*, 16(11):1077–1098, 2004.

[62] Blaise Gassend. Physical random functions. Master's thesis, Massachusetts Institute of Technology. Dept. of Electrical Engineering and Computer Science., January 2003.

[63] Blaise Gassend, Dwaine Clarke, Marten van Dijk, and Srinivas Devadas. Controlled Physical Random Functions . In *Proceedings of* 18^{th} *Annual Computer Security Applications Conference*, Silver Spring, MD, December 2002. Applied Computer Security Associates (ACSA).

[64] Blaise Gassend, Dwaine Clarke, Marten van Dijk, and Srinivas Devadas. Silicon physical random functions. In *Proceedings of the 9th ACM conference on Computer and communications security (CCS)*, 2002.

[65] Blaise Gassend, Dwaine Clarke, Marten van Dijk, and Srinivas Devadas. Delay-based circuit authentication and applications. In *Proceedings of the 2003 ACM Symposium on Applied Computing*, March 2003. Extended version in Concurrency and Computation: Practice and Experience.

[66] S. Goorden, M. Horstmann, A. Mosk, B. Skoric, and P. Pinske. Quantum-Secure Authentication with a Classical Key. *pre-print*, 2013. arXiv:1303.0142 [quant-ph].

[67] S. Graybeal and P. McFate. Getting out of the STARTing block. *Scientific American*, 261(6), 1989.

[68] Qian Guo, Thomas Johansson, and Carl Löndahl. Solving lpn using covering codes. In *Advances in Cryptology–ASIACRYPT 2014*, pages 1–20. Springer, 2014.

[69] Clemens Helfmeier, Christian Boit, Dmitry Nedospasov, and Jean-Pierre Seifert. Cloning Physically Unclonable Functions. In *IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, pages 1–6, 2013.

[70] Matthias Hiller, Dominik Merli, Frederic Stumpf, and Georg Sigl. Complementary IBS: Application Specific Error Correction for PUFs. In *IEEE Int. Symposium on Hardware-Oriented Security and Trust*. IEEE, 2012.

[71] Matthias Hiller, Michael Weiner, Leandro Rodrigues Lima, Maximilian Birkner, and Georg Sigl. Breaking Through Fixed PUF Block Limitations with Differential Sequence Coding and Convolutional Codes. In *Proceedings of the 3rd International Workshop on Trustworthy Embedded Devices*, TrustED '13, pages 43–54, 2013.

[72] Mark Van Hoeij, Jean-François Ragot, Felix Ulmer, and Jacques-Arthur Weil. Liouvillian solutions of linear differential equations of order three and higher. *Journal of Symbolic Computation*, 28(4):589–609, 1999.

[73] Johannes Hoffmann, Christian Hafner, Patrick Leidenberger, Jan Hesselbarth, and Sven Burger. Comparison of electromagnetic field solvers for the 3d analysis of plasmonic nanoantennas. In *SPIE Europe Optical Metrology*, pages 73900J–73900J. International Society for Optics and Photonics, 2009.

[74] D.E. Holcomb, W. Burleson, and K. Fu. Initial SRAM State as a Fingerprint and Source of True Random Numbers for RFID Tags. In *Proceedings of the Conference on RFID Security*, July 2007.

[75] D.E. Holcomb, W. Burleson, and K. Fu. Power-up SRAM State as an Identifying Fingerprint and Source of True Random Numbers. *IEEE Transactions on Computers*, 58(9):1198–1210, September 2009.

[76] TR Hopkins. On the sensitivity of the coefficients of padé approximants with respect to their defining power series coefficients. *Journal of Computational and Applied Mathematics*, 8(2):105–109, 1982.

[77] Nicholas J Hopper and Manuel Blum. Secure human identification protocols. In *Advances in cryptology: ASIACRYPT 2001*, pages 52–66. Springer, 2001.

[78] Gabriel Hospodar, Roel Maes, and Ingrid Verbauwhede. Machine Learning Attacks on 65nm Arbiter PUFs: Accurate Modeling poses strict Bounds on Usability. In *4th IEEE International Workshop on Information Forensics and Security (WIFS 2012)*, pages 37 – 42, 2012.

[79] J. H. Hubbard and B. E. Lundell. A first look at differential algebra.

[80] I. Kaplansky. *An Introduction to Differential Algebra.* Hermann, 1957.

[81] Irving Kaplansky. *An introduction to differential algebra*, volume 1251. Hermann, 1976.

[82] D. Karakoyunlu and B. Sunar. Differential template attacks on PUF enabled cryptographic devices. In *Information Forensics and Security (WIFS), 2010 IEEE International Workshop on*, pages 1–6, Dec 2010.

[83] Akitoshi Kawamura. Complexity of initial value problems. *Fields Institute Communications (to appear)*, 2010.

[84] Akitoshi Kawamura and Stephen Cook. Complexity theory for operators in analysis. *ACM Transactions on Computation Theory (TOCT)*, 4(2):5, 2012.

[85] Akitoshi Kawamura, Hiroyuki Ota, Carsten Rösnick, and Martin Ziegler. Computational complexity of smooth differential equations. In *Mathematical Foundations of Computer Science 2012*, pages 578–589. Springer, 2012.

[86] D. Kirovski. Anti-counterfeiting: Mixing the physical and the ditigal world. In A.-R. Sadeghi and D. Naccache, editors, *Towards Hardware-Intrinsic Security*, pages 223–233. Springer, 2010.

[87] P. Koeberl, Jiangtao Li, A. Rajan, and Wei Wu. Entropy loss in PUF-based key generation schemes: The repetition code pitfall. In *Hardware-Oriented Security and Trust (HOST), 2014 IEEE International Symposium on*, pages 44–49, May 2014.

[88] E.R. Kolchin. *Differential Algebra & Algebraic Groups*. Pure and Applied Mathematics. Elsevier Science, 1973.

[89] Joonho Kong, Farinaz Koushanfar, Praveen K. Pendyala, Ahmad-Reza Sadeghi, and Christian Wachsmann. PUFatt: Embedded Platform Attestation Based on Novel Processor-Based PUFs. In *ACM/IEEE Design Automation Conference (DAC), to appear*. ACM, June 2014.

[90] Jerald J. Kovacic. An algorithm for solving second order linear homogeneous differential equations. *Journal of Symbolic Computation*, 2(1):3 – 43, 1986.

[91] H Krawczyk, M Bellare, and R Canetti. HMAC: Keyed-Hashing for Message Authentication. 1997.

[92] Raghavan Kumar and Wayne Burleson. On design of a highly secure PUF based on non-linear current mirrors. In *2014 IEEE International Symposium on Hardware-*

Oriented Security and Trust, HOST 2014, Arlington, VA, USA, May 6-7, 2014, pages 38–43, 2014.

[93] 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 with identification and authentication applications. In *Proceedings of the IEEE VLSI Circuits Symposium*, 2004.

[94] Éric Levieil and Pierre-Alain Fouque. An improved lpn algorithm. In *Security and Cryptography for Networks*, pages 348–359. Springer, 2006.

[95] D. Lim, J. W. Lee, B. Gassend, G. E. Suh, M. van Dijk, and S. Devadas. Extracting secret keys from integrated circuits. *IEEE Trans. VLSI Syst.*, 13(10):1200–1205, 2005.

[96] Daihyun Lim. Extracting secret keys from integrated circuits. Master's thesis, Massachusetts Institute of Technology. Dept. of Electrical Engineering and Computer Science., May 2004.

[97] Vadim Lyubashevsky. The parity problem in the presence of noise, decoding random linear codes, and the subset sum problem. In *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques*, pages 378–389. Springer, 2005.

[98] R. Maes, P. Tuyls, and I. Verbauwhede. Low-Overhead Implementation of a Soft Decision Helper Data Algorithm for SRAM PUFs. In *Cryptographic Hardware and Embedded Systems (CHES)*, pages 332–347, 2009.

[99] Roel Maes, Pim Tuyls, and Ingrid Verbauwhede. Soft Decision Helper Data Algorithm for SRAM PUFs. In *Proceedings of the 2009 IEEE International Conference on Symposium on Information Theory - Volume 3*, ISIT'09, pages 2101–2105, 2009.

[100] Roel Maes, Anthony Van Herrewege, and Ingrid Verbauwhede. PUFKY: A Fully Functional PUF-based Cryptographic Key Generator. In *Proceedings of the 14th International Conference on Cryptographic Hardware and Embedded Systems*, CHES'12, pages 302–319, 2012.

[101] A. R. Magid. *Lectures on Differential Galois Theory.* University Lecture Series. American Mathematical Society, 1994.

[102] Ahmed Mahmoud, Ulrich Rührmair, Mehrdad Majzoobi, and Farinaz Koushanfar. Combined Modeling and Side Channel Attacks on Strong PUFs. Cryptology ePrint Archive, Report 2013/632, 2013.

[103] M. Majzoobi, A. Elnably, and F. Koushanfar. FPGA Time-bounded Unclonable Authentication. In *Information Hiding*, pages 1–16. Springer, 2010.

[104] M. Majzoobi and F Koushanfar. Time-Bounded Authentication of FPGAs. *IEEE Transactions on Information Forensics and Security*, 6(3):1123–1135, 2011.

[105] M. Majzoobi, F. Koushanfar, and M. Potkonjak. Lightweight secure PUFs. In *ACM/IEEE International Conference on Computer-Aided Design (ICCAD)*, pages 670–673, 2008.

[106] M. Majzoobi, F Koushanfar, and M. Potkonjak. Testing techniques for hardware security. In *IEEE International Test Conference (ITC)*, pages 1–10, 2008.

[107] M. Majzoobi, F. Koushanfar, and M. Potkonjak. Techniques for Design and Implementation of Secure Reconfigurable PUFs. *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, 2(1), 2009.

[108] Mehrdad Majzoobi, Ahmed Elnably, and Farinaz Koushanfar. FPGA Time-Bounded Unclonable Authentication. In *Information Hiding*, pages 1 – 16, 2010.

[109] A. Theodore Markettos and Simon W. Moore. The Frequency Injection Attack on Ring-Oscillator-Based True Random Number Generators. In *Cryptographic Hardware and Embedded Systems (CHES)*, pages 317–331. 2009.

[110] Dominik Merli, Dieter Schuster, Frederic Stumpf, and Georg Sigl. Semi-invasive EM attack on FPGA RO PUFs and countermeasures. In *Proceedings of the Workshop on Embedded Systems Security*, WESS, pages 2:1–2:9, 2011.

[111] JulianF. Miller, SimonL. Harding, and Gunnar Tufte. Evolution-in-materio: evolving computation in materials. *Evolutionary Intelligence*, 7(1):49–67, 2014.

[112] Kaizad Mistry, C Allen, C Auth, B Beattie, D Bergstrom, M Bost, M Brazier, M Buehler, A Cappellani, R Chau, et al. A 45nm logic technology with high-k+

metal gate transistors, strained silicon, 9 cu interconnect layers, 193nm dry patterning, and 100% pb-free packaging. In *Electron Devices Meeting, 2007. IEDM 2007. IEEE International*, pages 247–250. IEEE, 2007.

[113] Michael A Nielsen and Isaac L Chuang. *Quantum computation and quantum information.* Cambridge university press, 2010.

[114] Jason S Orcutt, Benjamin Moss, Chen Sun, Jonathan Leu, Michael Georgas, Jeffrey Shainline, Eugen Zgraggen, Hanqing Li, Jie Sun, Matthew Weaver, et al. Open foundry platform for high-performance electronic-photonic integration. *Opt. Express*, 20(11):12222–12232, 2012.

[115] Yossef Oren, Ahmad-Reza Sadeghi, and Christian Wachsmann. On the Effectiveness of the Remanence Decay Side-Channel to Clone Memory-Based PUFs. In *Cryptographic Hardware and Embedded Systems (CHES)*, pages 107–125. 2013.

[116] M. Orshansky. Physically unclonable functions based on non-linearity of sub-threshold operation, 2015. US Patent 8,938,069.

[117] Steven A Orszag and CM Bender. Advanced mathematical methods for scientists and engineers. *Mac Graw Hill*, 1978.

[118] P Packan, S Akbar, M Armstrong, D Bergstrom, M Brazier, H Deshpande, K Dev, G Ding, T Ghani, O Golonzka, et al. High performance 32nm logic technology featuring 2 nd generation high-k+ metal gate transistors. In *Electron Devices Meeting (IEDM), 2009 IEEE International*, pages 1–4. IEEE, 2009.

[119] T. N. Palmer. Towards the probabilistic earth-system simulator: a vision for the future of climate and weather prediction. *Quarterly Journal of the Royal Meteorological Society*, 138(665):841–861, 2012.

[120] R. S. Pappu, B. Recht, J. Taylor, and N. Gershenfeld. Physical one-way functions. *Science*, 297:2026–2030, 2002.

[121] Ravikanth Pappu. *Physical One-Way Functions.* PhD thesis, Massachusetts Institute of Technology, 2001.

[122] Yvan Paquot, Francois Duport, Antoneo Smerieri, Joni Dambre, Benjamin Schrauwen, Marc Haelterman, and Serge Massar. Optoelectronic reservoir computing. *Scientific reports*, 2, 2012.

[123] Z. Paral and S Devadas. Reliable and efficient PUF-based key generation using pattern matching. In *IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, pages 128–133, 2011.

[124] Sohan Purohit and Martin Margala. Investigating the impact of logic and circuit implementation on full adder performance. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 20(7):1327–1331, 2012.

[125] Dominik G Rabus. Ring resonators: Theory and modeling. *Integrated Ring Resonators: The Compendium*, pages 3–40, 2007.

[126] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM)*, 56(6):34, 2009.

[127] Michal Rewienski and Jacob White. A trajectory piecewise-linear approach to model order reduction and fast simulation of nonlinear circuits and micromachined devices. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 22(2):155–170, 2003.

[128] Joseph Fels Ritt. *Differential algebra*, volume 33. American Mathematical Soc., 1950.

[129] Lee A Rubel. Digital simulation of analog computation and church's thesis. *The Journal of Symbolic Logic*, 54(03):1011–1017, 1989.

[130] U. Rührmair. SIMPL Systems: On a Public Key Variant of Physical Unclonable Functions. Technical report, Cryptology ePrint Archive, International Association for Cryptologic Research, 2009.

[131] 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)*, pages 237–249. ACM, 2010.

[132] U. Rührmair, J. Sölter, F. Sehnke, Xiaolin Xu, A. Mahmoud, V. Stoyanova, G. Dror, J. Schmidhuber, W. Burleson, and S. Devadas. PUF Modeling Attacks on Simu-

lated and Silicon Data. *Information Forensics and Security, IEEE Transactions on*, 8(11):1876–1891, Nov 2013.

[133] U. Rührmair, M. Stutzmann, P. Lugli, C. Jirauschek, K. Müller, H. Langhuth, G. Csaba, E. Biebl, and J. Finley. Method and system for security purposes. European Patent Application Nr. EP 09 157 041.6, March 2009.

[134] Ulrich Rührmair. SIMPL Systems, Or: Can we construct cryptographic hardware without secret key information? In *International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM)*. Springer, 2011. Invited Paper, to appear.

[135] Ulrich Rührmair, Qingqing Chen, Martin Stutzmann, Paolo Lugli, Ulf Schlichtmann, and György Csaba. Towards Electrical, Integrated Implementations of SIMPL Systems. *Information Security Theory and Practices. Security and Privacy of Pervasive Systems and Smart Devices*, 6033:277–292, 2010.

[136] Ulrich Rührmair, Qingqing Chen, Martin Stutzmann, Paolo Lugli, Ulf Schlichtmann, and György Csaba. Towards electrical, integrated implementations of simpl systems. In *Workshop in Information Security Theory and Practice (WISTP)*, pages 277–292, 2010.

[137] Ulrich Rührmair, Srinivas Devadas, and Farinaz Koushanfar. Security based on Physical Unclonability and Disorder. In M. Tehranipoor and C. Wang, editors, *Introduction to Hardware Security and Trust*, chapter 4, pages 65–102. Springer, 2012.

[138] Ulrich Rührmair, Xiaolin Xu, Jan Sölter, Ahmed Mahmoud, Farinaz Koushanfar, and Wayne Burleson. Power and Timing Side Channels for PUFs and their Efficient Exploitation. Cryptology ePrint Archive, Report 2013/851, 2013.

[139] Ulrich Ruhrmair, Xiaolin Xu, Jan Solter, Ahmed Mahmoud, Mehrdad Majzoobi, Farinaz Koushanfar, and Wayne Burleson. Efficient power and timing side channels for physical unclonable functions. In *Cryptographic Hardware and Embedded Systems–CHES 2014: 16th International Workshop, Busan, South Korea, September 23-26, 2014, Proceedings*, volume 8731, page 476. Springer, 2014.

[140] Juan José Morales Ruiz. *Differential Galois theory and non-integrability of Hamiltonian systems*, volume 179. Springer, 1999.

[141] Rahul Sarpeshkar. Analog versus digital: extrapolating from electronics to neurobiology. *Neural computation*, 10(7):1601–1638, 1998.

[142] B. David Saunders. An implementation of kovacic's algorithm for solving second order linear homogeneous differential equations. In *Proceedings of the fourth ACM symposium on Symbolic and algebraic computation*, SYMSAC '81, pages 105–108, New York, NY, USA, 1981. ACM.

[143] Wil Schilders. Introduction to model order reduction. In *Model Order Reduction: Theory, Research Aspects and Applications*, pages 3–32. Springer, 2008.

[144] Wilhelmus HA Schilders, Henk A Van Der Vorst, and Joost Rommes. *Model order reduction: theory, research aspects and applications*, volume 13. Springer, 2008.

[145] Arnold Schönhage. *On the power of random access machines*. Springer, 1979.

[146] P. Sebbah, B. Hu, A. Z. Genack, R. Pnini, and B. Shapiro. Spatial-field correlation: The building block of mesoscopic fluctuations. *Phys. Rev. Lett.*, 88:123901, Mar 2002.

[147] Claude E Shannon. Mathematical theory of the differential analyzer. *J. Math. Phys. MIT*, 20:337–354, 1941.

[148] Hava Siegelmann. *Neural networks and analog computation: beyond the Turing limit*, volume 20. Springer Science & Business Media, 1999.

[149] M. F. Singer and Marius van der Put. *Galois Theory of Linear Differential Equations*. Springer, 2003.

[150] Saurabh Sinha, Greg Yeric, Vikas Chandra, Brian Cline, and Yu Cao. Exploring sub-20nm finfet design with predictive technology models. In *Proceedings of the 49th Annual Design Automation Conference*, pages 283–288. ACM, 2012.

[151] Carolyn Jane Smith and University of Waterloo. Faculty of Mathematics. *A discussion and implementation of Kovacic's algorithm for ordinary differential equations*. Faculty of Mathematics, University of Waterloo, 1984.

[152] Renée St Amant, Amir Yazdanbakhsh, Jongse Park, Bradley Thwaites, Hadi Esmaeilzadeh, Arjang Hassibi, Luis Ceze, and Doug Burger. General-purpose code acceleration with limited-precision analog computation. In *Proceeding of the 41st annual international symposium on Computer architecuture*, pages 505–516. IEEE Press, 2014.

[153] Herbert Stahl. The convergence of padé approximants to functions with branch points. *journal of approximation theory*, 91(2):139–204, 1997.

[154] Larry Stockmeyer. The complexity of approximate counting. In *Proceedings of the fifteenth annual ACM symposium on Theory of computing*, pages 118–126. ACM, 1983.

[155] Josef Stoer and Roland Bulirsch. *Introduction to numerical analysis*, volume 12. Springer Science & Business Media, 2013.

[156] Y. Su, J. Holleman, and B. Otis. A 1.6pJ/bit 96 (percent) Stable Chip ID Generating Circuit using Process Variations. In *IEEE International Solid-State Circuits Conference (ISSCC)*, pages 200–201, 2007.

[157] G. Edward Suh. *AEGIS: A Single-Chip Secure Processor.* PhD thesis, Massachusetts Institute of Technology. Dept. of Electrical Engineering and Computer Science., August 2005.

[158] G. Edward Suh and Srinivas Devadas. Physical unclonable functions for device authentication and secret key generation. In *ACM/IEEE Design Automation Conference (DAC)*, 2007.

[159] DanE. Tamir, NatanT. Shaked, WilhelmusJ. Geerts, and Shlomi Dolev. Combinatorial optimization using electro-optical vector by matrix multiplication architecture. In Shlomi Dolev and Mihai Oltean, editors, *Optical SuperComputing*, volume 5882 of *Lecture Notes in Computer Science*, pages 130–143. Springer Berlin Heidelberg, 2009.

[160] David J Thomson, Frederic Y Gardes, Jean-Marc Fedeli, Sanja Zlatanovic, Youfang Hu, Bill Ping Piu Kuo, Evgeny Myslivets, Nikola Alic, Stojan Radic, Goran Z Mashanovich, et al. 50-gb/s silicon optical modulator. *Photonics Technology Letters, IEEE*, 24(4):234–236, 2012.

[161] Max Tillmann, Borivoje Dakić, René Heilmann, Stefan Nolte, Alexander Szameit, and Philip Walther. Experimental boson sampling. *Nature Photonics*, 7(7):540–544, 2013.

[162] Johannes Tobisch and Georg T. Becker. On the Scaling of Machine Learning Attacks on PUFs with Application to Noise Bifurcation. In *Proceedings of RFIDSec 2015*, 2015.

[163] Seinosuke Toda. Pp is as hard as the polynomial-time hierarchy. *SIAM Journal on Computing*, 20(5):865–877, 1991.

[164] Lloyd N Trefethen. *Approximation theory and approximation practice*. Siam, 2013.

[165] N. Trefethen. *Approximation Theory and Approximation Practice*. Society for Industrial and Applied Mathematics, 2012.

[166] F. Koushanfar U. Rührmair, S. Devadas. *Security based on Physical Unclonability and Disorder*, chapter 5. unpub.

[167] Leslie G Valiant. The complexity of computing the permanent. *Theoretical computer science*, 8(2):189–201, 1979.

[168] Peter van Emde Boas. Handbook of theoretical computer science (vol. a). chapter Machine Models and Simulations, pages 1–66. MIT Press, Cambridge, MA, USA, 1990.

[169] Wolfgang Wasow. *Asymptotic expansions for ordinary differential equations*. Courier Dover Publications, 2002.

[170] Roscoe B White. *Asymptotic analysis of differential equations*. World Scientific, 2005.

[171] James H Wilkinson et al. The perfidious polynomial. 1984.

[172] C. Yang, G.X. Cui, Y.Y. Huang, L. Wu, H. Yang, and Y.H. Zhang. Performance of an embedded optical vector matrix multiplication processor architecture. *Optoelectronics, IET*, 4(4):159–164, 2010.

[173] M.-D. M. Yu, R. Sowell, A. Singh, D. M'Raihi, and S Devadas. Performance metrics and empirical results of a PUF cryptographic key generation ASIC. In *IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, pages 108–115, 2012.

[174] Meng-Day (Mandel) Yu and Srinivas Devadas. Secure and robust error correction for physical unclonable functions. *IEEE Design and Test of Computers*, 27:48–65, 2010.

[175] Meng-Day (Mandel) Yu, Matthias Hiller, and Srinivas Devadas. Maximum-likelihood decoding of device-specific multi-bit symbols for reliable key generation. In *IEEE International Symposium on Hardware Oriented Security and Trust, HOST 2015*, pages 38–43, 2015.

[176] Meng-Day (Mandel) Yu, David M'Raïhi, Richard Sowell, and Srinivas Devadas. Lightweight and Secure PUF Key Storage Using Limits of Machine Learning. In *Cryptographic Hardware and Embedded Systems (CHES)*, pages 358–373. 2011.