RICE UNIVERSITY

Indelible Physical Randomness for Security: Silicon, Bisignals, Biometrics

by

Masoud Rostami

A THESIS SUBMITTED
IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE

Doctor of Philosophy

APPROVED, THESIS COMMITTEE:

Farinaz Koushanfar, Chairman
Associate Professor of Electrical and Computer Engineering
Rice University

Edward Knightly
Professor of Electrical and Computer Engineering
Rice University

Dan S. Wallach
Professor of Computer Science
Rice university

Ari Juels
Professor of Jacobs Technion-Cornell Institute

HOUSTON, TEXAS

MAY, 2014

Abstract

Indelible Physical Randomness for Security: Silicon, Bisignals, Biometrics

by

Masoud Rostami

In this thesis, I investigate the nature and properties of several indelible physical randomness phenomena. I leverage these indelible statistical properties to design robust and efficient security systems. Three different phenomena are discussed in this thesis: randomness in biosignals, silicon chips, and biometrics. In the first part, I present a system to authenticate external medical device programmers to Implantable Medical Devices (IMDs). IMDs have now built-in radio communication to facilitate non-invasive reprogramming, but lack well-designed authentication mechanisms, exposing patients to the risks of over-the-air attacks and physical harm. Our protocol uses biosignals for authentication mechanism, ensuring access only by a medical instrument in physical contact with an IMD-bearing patient. Based on statistical analysis of real-world data, I propose and analyze new techniques for extracting time-varying randomness from biosignals and introduce a novel cryptographic device pairing protocol that uses this randomness to protect against attacks by active adversaries, while meeting the practical challenges of lightweight implementation and noise tolerance in biosignals readings. In the second part, unavoidable physical randomness of transistors is investigated, and novel robust and low-overhead authentication, bit-commitment, and key exchange protocols are proposed. It will be meticulously shown that these protocols can achieve resiliency against reverse-engineering and replay attacks

without a costly secure channel. The attack analysis guides us in tuning the parameters of the protocols for an efficient and secure implementation. In the third part, the statistical properties of fingerprint minutiae points are analyzed and a distributed security protocol will be proposed to safeguard biometric fingerprint databases based on the developed statistical models of fingerprint biometric.

Acknowledgements

I would like to thanks my Lovely wife, Golnaz, for her warm support during these past few years. Without her, none of this would have been possible. I would like to thank my adviser, Prof. Farinaz Koushanfar, for her astonishing patience and valuable support during my projects. I am grateful for the opportunity of working with a world-class scientist, Dr. Ari Juels, and learning from him. I would also like to thank my committee members for their valuable time and comments on this thesis.

Contents

List of Figures

List of Tables

Chapter 1

Indelible physical randomness for security

In this thesis, I discuss using indelible physical randomness of chips, biometrics, and biosignals to achieve robust security with acceptable implementation overhead.

In the next chapter, we begin our discussion by investigating the problem of securing implantable medical devices (IMDs) against eavesdroppers and hackers. Implantable Medical Devices (IMDs) are being embedded increasingly often in patients' bodies to monitor and help treat medical conditions. To facilitate monitoring and control, IMDs are often equipped with wireless interfaces. While convenient, wireless connectivity raises the risk of malicious access to an IMD that can potentially infringe patients' privacy and even endanger their lives. We review proposed approaches to the access-control problem for IMDs, including the problem of secure pairing (and key distribution) between an IMD and another device, such as a programmer. (We also treat related technologies, such as body-area networks.) We describe some limitations of well-conceived proposals and reveal security weaknesses in two proposed cryptographic pairing schemes. Our intention is to stimulate yet more inventive and rigorous research in the intriguing and challenging areas of IMD security and medical-device security in general.

To achieve our own of goal of a secure IMD communication, we start of by analyzing

the statistical characteristics of ECG, which is the most accessible biosignal. We present Heart-to-Heart (H2H), a system to authenticate external medical device controllers and programmers to Implantable Medical Devices (IMDs). IMDs, which include pacemakers and cardiac defibrillators, are therapeutic medical devices partially or wholly embedded in the human body. They often have built-in radio communication to facilitate non-invasive reprogramming and data readout. Many IMDs, though, lack well designed authentication protocols, exposing patients to over-the-air attack and physical harm. H2H makes use of ECG (heartbeat data) as an authentication mechanism, ensuring access only by a medical instrument in physical contact with an IMD-bearing patient. Based on statistical analysis of real-world data, we propose and analyze new techniques for extracting time-varying randomness from ECG signals for use in H2H. We introduce a novel cryptographic device pairing protocol that uses this randomness to protect against attacks by active adversaries, while meeting the practical challenges of lightweight implementation and noise tolerance in ECG readings. Finally, we describe an end-to-end implementation in an ARM-Cortex M-3 microcontroller that demonstrates the practicality of H2H in current IMD hardware.

In the third chapter, we turn our attention to the inherent physical randomness of chips, and leverage them to implement robust and low-overhead authentication, key exchange, and commitment protocols that are resilient against reverse-engineering attacks. We will use novel circuit entities called physical unclonable functions (PUFs) that transform the indelible randomness of chips to digital outputs. Our proposed protocols are executed between a party with access to a physical PUF (Prover) and a trusted party who has access to the PUF compact model (Verifier). The proposed protocols do not follow the classic paradigm of exposing the full PUF responses or a transformation of them. Instead, random subsets of PUF response strings are sent to the Verifier so the exact position of the subset is obfuscated for the third-party channel observers. Authentication of the responses at the Verifier side is done by matching the substring to the available full response string; the index of the matching point is the actual obfuscated secret (or key) and not the response substring itself. We perform a thorough analysis of resiliency of the protocols against vari-

ous adversarial acts, including machine learning and statistical attacks. The attack analysis guides us in tuning the parameters of the protocol for an efficient and secure implementation. The low overhead and practicality of the protocols are evaluated and confirmed by hardware implementation.

In the last chapter of this thesis, I address the issue of persistent attacks and hacks against supposedly secure servers biometric data-storage servers. We will discuss a reliable distributed fingerprint storage protocol that has an acceptable overhead. The protocol is inspired by the Honeyword password protocol.

Chapter 2

Indelible biosignal randomness for security implantable medical devices

2.1 introduction

Implantable Medical Devices (IMDs) apply continuous monitoring and automatic therapies to the treatment of chronic medical disorders. Implanted either partially or fully in patients' bodies, IMDs are often sophisticated devices containing batteries, embedded CPUs, radios, sensors, and actuators. As clinical trials validate IMDs' efficacy [44] and as IMDs treat a broadening range of disorders, their use is growing. For instance, in the United States, over 100,000 patients a year receive implantable cardioverter defibrillators (ICDs) [40], which detect dangerous heart rhythms and administer electric shocks to restore normal activity. Other IMDs include pacemakers, neurostimulators, and implantable drug pumps. Over the past few decades, IMDs have greatly improved patient care, quality of life, and life expectancy. Next-generation IMDs will provide even more benefit, as they improve existing therapies and enable new ones.

Powered IMDs generally contain radios for communication with external devices called *commercial device programmers* that can reprogram IMDs and extract patient data from them. Such wireless communication permits safe, non-invasive access to IMDs. Wireless

interfaces contribute significantly to the utility and successful deployment of IMDs, as they enable convenient and non-invasive control, monitoring, and maintenance of the IMD using a control device typically known as a "programmer." A drawback to such wireless access, though, is its inherently open nature, which raises IMDs susceptibility to over-the-air adversarial threats ranging from eavesdropping to unauthorized access and control. IMD manufacturers are subject to strict safety and reliability requirements, and the safety of IMDs, including the problem of unexpected failures, has been a subject of ongoing research [70]. IMDs are not subject, however, to similar standards around *logical security* and *access control*. In many cases, the approach has been protection of IMDs through security-by-obscurity: The main barrier to unauthorized access is no more than secret and proprietary design. Seminal work by Halperin et al. [42], for example, exposes design flaws in a common ICD that enable attackers to seize unauthorized control wirelessly, and potentially harm victims.

Researchers have consequently demonstrated a range of practical attacks, some executed remotely over the air, permitting unauthorized access to IMDs such as cardiac defibrillators and insulin pumps [42, 61]. These attacks enable an adversary to eavesdrop on IMD communications and in some cases emulate a programmer and modify the therapies applied by IMDs, potentially threatening patient privacy and even patients' health and lives. While there are no documented examples of such attacks "in the wild," the need for better secured access control in IMDs (and robust logical security more generally) is urgent and clear.

Our work here addresses the tension between two critical requirements for IMDs. On the one hand, IMDs must offer reasonably permissive access-control policies when life-threatening medical events occur. Emergency responders may need to reprogram IMDs or extract patient data from them, and shouldn't incur fatal treatment delays by contacting care providers for device-specific keys or passwords. On the other hand, overly loose access-control policies expose IMDs to unauthorized wireless access, such as those illustrated by the Halperin et al. attack, that can physically harm patients or expose their medical

data [42].

One major challenge in designing good access-control and other security mechanisms for IMD is their severely constrained resources. IMDs, like other portable electronics, have limited available battery energy. For IMDs, the situation is particularly problematic, as battery replacement usually entails invasive surgery and removal/replacement of the IMD or its components. Additionally, the desire for minimally invasive implantation favors small form-factors for IMDs, further limiting battery size. Remote power delivery and energy scavenging, although promising, are currently available in only a very limited set of applications.

Three ongoing trends suggest that energy challenges will persist for IMDs. First, the devices are getting increasingly complex and power-hungry due to demand for new, sophisticated therapeutic and monitoring functionality. Power requirements are even outstripping the benefits of Moore's Law and low-power design techniques, as with smart-phones. Second, IMDs are collecting ever more data as new sensors are added to monitor patient health. Transmitting sensor data from an IMD involves wireless communication, which is power intensive. Third, well designed security protocols, including authentication and code verification require the use of cryptography, and cryptographic primitives are notoriously computation- and power-intensive.

A second major challenge in securing IMDs is the tension between the demands of reliable access on the one hand, and protection against access by an adversary or unauthorized entity on the other. In an emergency, medical personnel may need to monitor or reprogram a patient's IMD immediately and thus access it with as little impediment as possible. But an IMD that can be accessed too easily may be vulnerable to eavesdropping on its data transmissions or tampering with its operation.

The conflicting requirements of security, reliability, and usability in IMDs have given rise to an important and vigorous line of research. Halperin et al. [43] first discussed the security and privacy challenges arising from the resource constraints and inflexibility of existing IMD designs, and highlighted fundamental tensions among privacy, security, safety,

and utility in IMDs. Increased networking of embedded devices and the emergence of pervasive health care technologies have also motivated closely related security and privacy research for general sensor networks and body-sensor networks (e.g., [120, 127]), health-care information technology (e.g., [82]), and patient health data (e.g., [47]).

2.1.1 Our proposed solution based on biosignal randomness

Our solution is a system called *Heart-to-Heart* (H2H). H2H implements a simple access-control policy for IMDs that we call "touch-to-access": A medical instrument (e.g., commercial device programmer), which we call generically a *Programmer*, obtains access to a patient's IMD if and only if it has significant physical contact with the patient's body. An important facet of touch-to-access is *forward security*. Authentication to the IMD lapses once the instrument loses physical contact with the patient.

Touch-to-access offers a practical and effective balance between the competing access requirements of permissiveness in emergencies and resistance to attacks. The policy is also common sense: Physical access to a patient means the ability to harm or cure.

H2H enforces a touch-to-access policy using a time-varying biometric, often called a *physiological value* (PV). When a Programmer seeks access to an IMD, it initiates an authentication session. The IMD takes a reading α of the PV; at the same time, the Programmer takes its own reading β. If β is "nearly equal" to α, then the Programmer obtains access to the IMD. ("Near equality," as we explain later, is needed because PV readings are noisy.)

The H2H architecture can in principle rely on any PV, but we focus here specifically on use of the waveform produced by the heart, known as an ECG (electrocardiogram). Thus H2H is well suited for authentication to cardiac IMDs such as ICDs and pacemakers, today the largest class of powered IMDs. In principle, though, H2H can work with *any* IMD equipped to measure ECG anywhere in the body, not just cardiac devices. As we show, suitably processed ECG samples effectively constitute a low-bandwidth stream of random bits well suited to forward-secure authentication.

Briefly, then, in H2H a Programmer and IMD take independent, time-synchronous ECG readings. The IMD compares the two results to enforce a touch-to-access policy on wireless access by the Programmer.

2.1.2 Security model and goals

The security goals for a device architecture naturally depend upon the participating trustworthy entities and the motives, access, capabilities, and resources of a potential adversary. IMDs typically communicate with a Programmer and potentially with other IMDs or outside-the-body medical devices. The adversary of main concern in these settings is one that acts remotely, over the air, against the IMD's network. Given the open nature of wireless networks, such an adversary may be "active," meaning that it has complete control of the network. Such an adversary can replay, modify, forge, drop, and jam message within the network at will. We can (at least in some cases) assume, however, that one side of the communication, the IMD, is not directly accessible to the adversary, as it is implanted in the body.

Again, a main objective is to allow a Programmer to gain logical access to an IMD while an adversary can't feasibly do so. An obvious approach would be to authenticate an entity communicating with an IMD using a predetermined key or password. The main obstacle to this approach is a fundamental challenge of cryptography: Key distribution. It's impractical to ensure that all valid programmers and/or medical personnel (in medical settings around the world) have valid keys but an adversary can't feasibly gain access to one.

A somewhat more flexible approach to key distribution is the use of public-key cryptography. The TLS protocol, which is ubiquitous on the Internet, relies upon the distribution of public keys to servers. A global public-key infrastructure (PKI) permits designated authorities to certify these public keys, ensuring a binding of the public key to a suitable server identity (domain name). Building a PKI for all medical programmers worldwide—and adequately securing all of their private keys—would be a formidable and probably impractical

effort. Recent breakdowns in the trustworthiness of certificate issuance for the Internet, e.g., [136], warn in general of the challenges of constructing sound PKIs.

Recent research on access control and authentication for IMDs and BANs has mainly focused on approaches in which Programmer authorization is determined as a function of *physical access or proximity*. As we explain, there are a number ways to determine whether a Programmer is in suitable proximity to an IMD. The problem often boils down, however, to one of *key-agreement* or *pairing*. Ideally, only an authorized Programmer should be able to establish a (secret) cryptographic key with an IMD; this key enables the establishment of a secure (confidential and integrity-protected) channel between the two devices.

Thus we focus here mainly on proposed approaches to IMD access control through key agreement, which may occur directly between a Programmer and IMD or may be mediated by an additional trusted device carried by a patient. While there are several sound and well-conceived proposed approaches, none in our view provides a fully satisfactory balance of utility and security. Achieving rigorous security guarantees can also be quite challenging: We present attacks against two such proposed protocols, IMDGuard [132] and OPFKA [46]. Thus, the challenge of good IMD access-control remains an important open research problem.

2.1.3 Challenges and contributions

Several previous schemes have sought to perform ECG-based pairing with IMDs like H2H, but have had serious shortcomings. Most notably, previous schemes have relied on cryptographic pairing protocols without rigorous adversarial modeling or security analysis. As a result, two of the most recent of them [46, 132] were shown in a 2013 [100] to have serious cryptographic flaws.

Thus designing a practical system such as H2H with rigorous security assurances has effectively remained an open problem, one that raises several technical challenges.

The first challenge is demonstrating that ECG is a suitable PV for authentication. H2H derives a *key source* from the patient's ECG signal, a sequence of key bits that authenti-

cate a Programmer to an IMD. Secure touch-to-access authentication requires that the key source be *truly random*, ideally that constituent bits have high entropy and are statistically independent from one another and over time. The key source is then hard for an attacker to guess without physical access to the patient and also ensures forward-security, i.e., that old key source bits don't reveal future ones.

Previous work has explored the statistical properties of ECG waveforms for key generation, but not the important impact of read error rates on authentication false positive and false negative rates. We present experiments on real patient ECG data showing that it's possible (with errors) to extract roughly four truly random and statistically uncorrelated bits from the ECG wave corresponding to a single heartbeat. Collection over a 15-second interval suffices for strong Programmer authentication (a false acceptance rate of about 2.7×10^{-9} and false rejection rate of 10^{-4}).

We also introduce an optimal scheme in H2H for testing PV validity, i.e., testing $\alpha \approx \beta$. Our scheme relies on the Neyman-Pearson Lemma, rather than naïvely on Hamming distance, as in previous work.

Good statistical properties, however, don't ensure that ECG can enforce the touch-to-access policy in H2H. Recently developed systems can read cardiac rhythms *remotely* via videocamera, and even accurately measure a patient's pulse. (Skin color changes subtly with cardiac rhythms.) In chapter 2.5.3, we very briefly report our implementation and test results for the best known of these systems [92] and show that it doesn't reveal statistically significant information about the ECG key source used by H2H.

Given a good key source, a *cryptographic pairing protocol* is needed between the Programmer and IMD. Two features of H2H make cryptographic pairing a challenge. First, when the Programmer and IMD synchronously sample the key source, their respective readings β and α are *noisy*: Often $\beta \approx \alpha$, but exact equality $\beta = \alpha$ isn't obtained. Cryptographic tools such as password-authenticated key agreement (e.g., [11]) require exact equality, while error-tolerant ones, e.g., [30, 49], sacrifice entropy needlessly in our setting here.

Second, the IMD has tight *computational and power constraints*. Microcontrollers in common use for IMDs today can perform only lightweight cryptography. As IMDs are long-lived devices (with an average lifetime today of five to seven years [31]), and battery replacement requires surgical intervention, power conservation is essential. H2H can protect new IMDs as well as legacy in-vivo IMDs with upgradable firmware, as long as the H2H implementation meets the IMD's limited memory and computational resources.

We present a new pairing protocol that exploits the fact that key source bits are statistically uncorrelated, and thus that we can treat α and β as one-time authentication values. We demonstrate its security in a strong adversarial model that includes man-in-the-middle attacks, such as the jam-and-replay attacks feasible in a wireless environment.

Our H2H pairing protocol requires only a low-exponent RSA encryption (tens of modular multiplications) and a few AES invocations and hash computations by the IMD. We demonstrate a full implementation of H2H on an ARM Cortex-M3 processor.

In summary then, our contributions are:

- *Statistical characterization of ECG for authentication:* Using real-world ECG measurements [84], we experimentally quantify the extractable entropy in ECG signals. (We demonstrate use of the Neyman-Pearson Lemma to achieve optimal use of this randomness.)

- *Cryptographic pairing protocol:* We present a novel, lightweight, noise-tolerant cryptographic scheme for Programmer-to-IMD pairing in H2H. We formalize an adversarial model and outline proofs of security.

- *Implementation:* We describe a full implementation of H2H in an ARM Cortex-M3 processor, reporting resource requirements such as code size and power consumption, and demonstrating the feasibility of H2H for use in contemporary IMDs.

2.1.4 Organization

We introduce our statistical model for ECG waveforms and operational, trust, and adversarial models for H2H in chapter 2.4. A detailed statistical analysis of real-world ECG waveforms and our proposed entropy extraction techniques follows in chapter 2.5. We specify the cryptographic device-pairing protocol for H2H in chapter 2.6. We describe an implementation of H2H in chapter 2.8. We review related work in chapter 2.2, and conclude in chapter 5 with a discussion of future work. Chapter 2.7 outlines a security analysis of the H2H device-pairing protocol and detailed calculation of protocol parameters.

2.2 Related Work

Several early research and development efforts in medical electronics have addressed safety and reliability of IMD devices, particularly the problem of unexpected failures [70]. Increased networking of embedded devices and emergence of pervasive healthcare technologies motivated security and privacy investigations for general sensor networks and body sensor networks [33, 53, 115, 120, 127].

Halperin et al. [43] first discussed the security and privacy challenges caused by resource constraints and inflexibility in existing IMD designs, and highlighted fundamental tensions among privacy, security, safety, and utility. Fu [34] argued that improving IMD security requires a balance between technology and regulation.

Halperin et al. [42] gave the first systematic and pragmatic security analysis of a real commercial IMD, an implantable cardiovascular defibrillator (ICD). They showed that these devices are susceptible to attacks by malicious programmers that breach patient privacy and, even more seriously, can effect changes to data and functioning, potentially harming patients. Their work highlighted the pressing need for authentication of programmers to IMDs.

Programmer-to-IMD authentication, as noted above, is straightforward if the Programmer and IMD share a preexisting key. (Authentication tokens have the same simplicity, but

similar risks of IMD inaccessibility with credential loss.) Authentication without a pre-established relationship, as treated by H2H, is more challenging. There are several broad technical approaches bear comparison with H2H.

2.2.1 Key distribution by intra-body signaling

One idea for sharing a secret key between an IMD and Programmer is to generate the key in the IMD and transmit it to the Programmer through the human body itself. This approach requires that the Programmer be in close enough proximity to receive the key; generally, it may make physical contact with a patient. The critical security assumption is that an adversary must operate at a distance from the patient too great to intercept a key; the minimum required distance for such assurance depends on the specific scheme. We now briefly review three proposed intra-body carriers of IMD secret keys: acoustic, electric, and electromagnetic signals.

Acoustic broadcasting. Halperin et al. [42] proposed a scheme in which an implanted piezo device generates a random key and emits it acoustically. The method results in a rather fast key agreement, requiring only 400ms to transmit a 128-bit key. A serious drawback, however, is the requirement for special implantation of the piezo device. This implantation must be at a depth of 1 cm or so from the skin, ruling out incorporation into deep-body IMDs, such as Implantable Cardioverter Defibrillators (ICDs). The electronic circuits that produce acoustic signals can be shielded with a Faraday cage against electromagnetic interception. An adversary can potentially resort to eavesdropping on acoustic emanations, however, to attack the system. Acoustic eavesdropping of this kind is not well studied and merits further investigation.

This approach is complementary to H2H. It results in faster key agreement, needing a mere 400ms to emit a 128-bit key. A serious drawback, however, is that it requires special implantation of the piezo device. This implantation must be at a depth of 1 cm or so from the skin, ruling out incorporation into deep-body IMDs, such as ICDs. Additionally, eavesdropping on acoustic emanations isn't a well studied security problem.

Electric and Electromagnetic broadcasting. Zimmerman [137, 138] proposed transmitting information through the human body using a pico-amp electric current, in effect using the body as a low-frequency ($\approx$ 1MHz) electrical carrier. Chang et al. [18] discussed securing body area networks (BANs) by distributing a secret key using electrical currents below the action potential of human cells. They used empirical data to analyze the characteristics of the human body as a communication medium. They estimated 0.469 bits per hour as a lower bound on the bandwidth achievable with their proposed method. This is unacceptably slow, of course, for practical IMD key establishment.

In general, key distribution by intra-body communication has the potential to combine strong security against eavesdropping at a distance with minimal power consumption. The actual resistance of such methods to eavesdropping has not been well studied, however. These methods also require approval from government regulators that, to the best of our knowledge, has not yet been granted in the United States even for trial use.

2.2.2 Distance bounding

A promising approach, by Rassmussen et al. [97], uses ultrasound-based distance bounding to authenticate Programmer access to an IMD, achieving an access policy of proximity similar to those that use intra-body key transmission or key establishment using PVs. Their system requires RF shielding, however, amplifying the engineering complexity of an IMD. It also relies on RF communication. For some IMDs, e.g., brain implants, RF antennas are of prohibitive length, and alternatives, e.g., infrared, are preferred. Distance-bounding protocols' security models have also historically proven fragile (see, e.g., [22, 95]). Finally, this approach uses ultrasound transmission, which usually requires more power than RF transmission.

2.2.3 Jamming

Another approach to establishing a secure channel between an IMD and a Programmer (or other external device) is to make use of a trusted device to intermediate access to the IMD. This trusted device can be external to a patient's body, and thus well resourced. It can shield the IMD from unauthorized attempts at access and even potentially jam malicious ones.

The idea of blocking inappropriate access to an IMD was first proposed in [27] via a device called a Communication Cloaker. The idea saw a follow-up exploration by Gollakota et al. [39]. Their proposed device, called a *shield*, is worn near the body and used to authenticate / mediate Programmer (or other) communications with the IMD. Helpfully, a shield doesn't require modification of existing IMDs. It protects communications with the IMD using a full duplex radio device acting as a jammer-cum-receiver. It simultaneously listens to and jams IMD messages as appropriate, as well as unauthorized Programmer commands.

IMDGuard [132] is a similar method for IMD protection involving a third party, high-powered device worn externally, called a Guardian. The Guardian authenticates programmers on behalf of the IMD. Like H2H, it requires special-purpose IMD functionality.

Shen et al. [118] have recently proposed a smart jamming technique in which the shield jams the communication channel intermittently; a trusted Programmer knows in advance the intervals in which the channel is clear, and can thus communicate with the IMD. With this solution, the patient has the option of keeping the shield active even during Programmer communication with IMDs.

These approaches have the advantage of being compatible with legacy IMD, so they can be applied seamlessly to the currently deployed devices. A drawback, however, is that jamming, when employed to counteract attacks as in [39, 132], can disrupt the communications of other RF devices and violate laws regarding radio interference.

While these devices provide more general functionality, H2H has several advantages over them: (1) H2H doesn't require an external device, which is a burden on patients and

increases the risks of system failures and unreliability; (2) H2H doesn't require jamming, which, as employed to counter attacks in [39, 132], can interfere with other RF devices and potentially lead to legal complications.

While the shield has the benefit of legacy compatibility, we note that a growing number of legacy IMDs are built using programmable microcontrollers with in-vivo upgradable firmware, allowing an upgrade to the H2H protocol as long as its (lightweight) resource requirements are satisfied.

2.2.4 Key generation using biosiganls randomness

The idea of extracting secret keys from physiological values (PVs) to secure IMDs was first suggested in [20]. Numerous works subsequently used the randomness in ECG IPIs for IMD authentication, e.g., [6,21,93,124]. None, however, provided a rigorous entropy or protocol-security analysis. (In fact, the motivation for PV-derived keys in BANs is unclear. To pair *user-controlled devices* in *non-emergency settings*, even device passwords would seem practical.)

PVs such as Electrocardiograph (ECG) and Electroencephalography (EEG) signals are suitable candidates for key generation because they provide continuous sources of true randomness. In other words, these PVs may be viewed as entropy sources inside the human body that constantly generate and broadcast (unpredictable) random bits. The randomness of PVs has been documented in an extensive body of medical literature [38, 119, 135].

Due to its availability throughout the body and its ease of measurement, the most frequently proposed PV for securing IMDs has been the ECG signal, the electrical signal associated with the activity of the heart.

A number of challenges need to be addressed to achieve practical and secure use of PVs in key agreement. One obstacle is that PV readings are highly sensitive to probe locations on the body and to environmental conditions. Chang et al. [18] assert that if a transformation of the full ECG signal is used for authentication, as suggested in [125, 126], then the PV readings may be so noisy, given a poorly placed probe, that they can

be decoded as effectively at a distance by an adversary as by a Programmer with physical contact. The full ECG signal cannot be consistently decoded because the shape of the associated waveform is subject to distortion. Time intervals between specific waveform features, however, can in fact be reliably measured from nearly anywhere in the body. One such feature is the prominent R-Peak of the ECG signal: The time between two R-peaks, which is equal to the heartbeat duration, is essentially invariant to the positioning of probes on the body.

It has been shown that if the heartbeat duration is appropriately quantized, some of its least significant bits are truly random [6, 21, 93, 124]. These random bits may differ across probe points on the body due to measurement noise, limiting their naïve use as key bits shared between an IMD and Programmer. To address the challenge of noisy key sources, several methods have been advanced in previous work. For example, Xu et al. [132] proposed the IMDGuard protocol to securely pair an IMD with an external device using noisy ECG data to construct a key. IMDGuard looks to establish a persistent, cryptographic-strength key under non-emergency conditions. Unfortunately, the IMDGuard pairing protocol lacks a rigorous security analysis; Section 2.3 describes a man-in-the-middle attack that reduces its effective key length and hence its claimed level of security.

Another possible approach for securely extracting a cryptographic key from noisy PV readings is to use "fuzzy" cryptographic primitives, e.g., [30, 51]. Some proposed scheme use the "fuzzy vault" construction in [51] or variants thereof to authenticate devices in a body-area network [62, 125, 126]. Recently, Hu et al. [46] proposed a variant algorithm for PV-based key-agreement, called OPFKA, that is designed to reduce the storage costs associated with fuzzy vaults. OPFKA, however, lacks a rigorous security analysis and, as we explain in Section 2.3, has notable security weaknesses. The design of a reliable PV-based IMD key-agreement protocol with rigorously analyzed security properties, low power consumption, and a minimal hardware footprint remains a significant open research problem.

Most similar to H2H are two schemes: A protocol in IMDGuard for pairing the

Guardian with an IMD and a generic body-area network pairing protocol called OPFKA (Ordered-Physiological-Feature-based Key Agreement) [46]. Like H2H, these protocols make use of ECG measurements to authenticate a Programmer to an IMD. As noted above, however, both protocols lack rigorous security analysis and they have serious security flaws [100, 116]. (We also note that the reported hardware implementation overhead for the IMDGuard protocol greatly exceeds that of H2H.)

Compared with previous PV-based authentication schemes in general, H2H is the first work that: (1) Includes a statistical analysis of the full stochastic ECG waveform to demonstrate bit independence over time; (2) Quantifies and uses the individual error rates of the high-grade random bits to distinguish between honest and adversarial Programmers in an optimal way (via the Neyman-Pearson Lemma); and (3) Offers a formally analyzed PV-based cryptographic device pairing protocol.

2.3 Security Flaws of two recent biosignal based proposals

We now describe security weaknesses in two proposed protocols for authenticated key-agreement between devices in a body-area network. One is the setup protocol for the IMD-Guard system [132], which pairs a protective device called a "Guardian" with an IMD. The other, OPFKA (Ordered-Physiological-Feature-based Key Agreement), is a generic body-area network pairing protocol [46]. Both IMDGuard and OPFKA rely on ECG measurements as a common source of entropy to establish shared secret keys. While terminology differs across papers, and some involve devices in BANs, we continue to refer to devices generically as the Programmer and IMD.

2.3.1 Attack on IMDGuard

We briefly describe the IMDGuard scheme for key agreement between a Programmer (in IMDGuard, the Guardian) and IMD. We then show a simple man-in-the-middle attack that reduces the effective key length from 129 bits to 86 bits.

IMDGuard key-agreement protocol The Programmer and IMD each measure ECG data in a succession of four-bit blocks. Let $\alpha = (\alpha_1, \alpha_2, \alpha_3, \alpha_4)$ and $\beta = (\beta_1, \beta_2, \beta_3, \beta_4)$ denote respective measurements of one such block by the IMD and Programmer. As these readings are noisy, IMDGuard includes the following noise-reducing "reconciliation" scheme for extraction of key material.

In Round 1, the Programmer and IMD exchange parity bits: The IMD sends $\alpha_1 \oplus \alpha_2 \oplus \alpha_3 \oplus \alpha_4$, while the Programmer sends $\beta_1 \oplus \beta_2 \oplus \beta_3 \oplus \beta_4$, where $\oplus$ denotes XOR. If the parity bits agree, the two devices accumulate the first three bits as key material. (They discard a bit to compensate for the one bit leaked by parity-symbol disclosure.) Once 43 blocks pass the parity check, the two sides each possess 129 bits of key material. They hash their respective key material to generate check values h_α and h_β, which they then exchange. The Programmer compares these check values and sends an `accept` message if $h_\alpha = h_\beta$, and a `reject` message otherwise.

Round 2 takes place if (and only if) the Programmer determines that $h_\alpha \neq h_\beta$ (`reject`). In Round 2, the IMD transmits $\alpha_3 \oplus \alpha_4$ and the Programmer, $\beta_3 \oplus \beta_4$. If these two bits agree, the IMD retains α_2 and α_3 as key material, while the Programmer retains β_2 and β_3. I.e., the first bit of each block is discarded to compensate for parity-bit leakage. Further blocks are read and reconciled as needed. When enough bits have accumulated, check values are again compared. (The authors assert a Round 3 is never required.)

Man-in-the middle attack A man-in-the-middle adversary Adv can do the following. Adv allows the IMD and Programmer to proceed normally with parity-bit exchange in Round 1. Suppose that $h_\alpha = h_\beta$ (as happens with high probability). Adv makes two message substitutions at the end of the round: (1) Adv substitutes a random value for the check value h_α transmitted by the IMD, causing the Programmer to send a Round-1 `reject` message and proceed to Round 2 and (2) Adv substitutes an `accept` message for the Programmer's `reject` message, causing the IMD to terminate the protocol with the key established in Round 1.

The Programmer thus proceeds with Round 2. It transmits a second parity bit ($\beta_3 \oplus \beta_4$)

for each block from Round 1; at the same time, Adv simulates Round-2 parity-bit transmissions by the IMD. Adv intercepts Programmer parity-bit transmissions to recover an additional bit of information for each block. (For a given block α, the IMD uses $(\alpha_1, \alpha_2, \alpha_3)$ as key bits. Adv learns $\alpha_1 \oplus \alpha_2$.)

While the resulting effective key length of 86 bits is an infeasible target for brute-force attack today, this attack demonstrates a serious design weakness in IMDGuard.

2.3.2 Attack on OPFKA

In OPFKA, the IMD and Programmer each perform local ECG readings on a human subject over the same interval of time. They translate these readings into a temporally ordered sequence of "features," short (e.g., 12-bit) values. The two devices exploit overlap in their respective feature sequences to construct a shared secret key κ, much as with IMDGuard.

OPFKA adopts a different approach than IMDGuard, however, to specify this overlap. In OPFKA, the Programmer transmits its features obscured with spurious *chaff* values to the IMD in what is called a *coffer*. The IMD indicates to the Programmer those feature values in the coffer that lie in its own feature sequence. Each device, then, can determine the intersection of their two respective feature sequences which is used to construct the shared key κ.

Here is a more detailed specification of the protocol. For simplicity, we assume 12-bit features, one option in OPFKA. We omit protocol parameters and messages not germane to our analysis. For clarity, we also change some of the original notation for OPFKA.

1. **Feature reading:** Each device reads a sequence of N features. (In OPFKA, $N = 30$.) Let $\tilde{F}^{imd} = \{\tilde{f}_0^{imd}, \tilde{f}_1^{imd}, \ldots \tilde{f}_{N-1}^{imd}\}$ be the IMD's features, in temporal order, and $\tilde{F}^{pro} = \{\tilde{f}_0^{pro}, \tilde{f}_1^{pro}, \ldots \tilde{f}_{N-1}^{pro}\}$, the Programmer's.

2. **"Hashing":** Feature values in $\tilde{F}^{imd}$ and $\tilde{F}^{pro}$ are mapped into 20-bit feature values via a "hash" function $H : \{0,1\}^{12} \rightarrow \{0,1\}^{20}$. Let $F^{imd} = \{f_0^{imd}, f_1^{imd}, \ldots f_{N-1}^{imd}\}$ be the resulting set of feature values for the IMD and F^{pro} similarly for the Programmer.

3. **Coffer transmission:** The Programmer randomly selects M chaff features $F' = \{f'_0, f'_1, \ldots, f'_{M-1}\}$, where $f'_i \in_R \{0,1\}^{20} - F^{pro}$. It randomly permutes elements in $C = F \bigcup F'$ and sends the resulting *coffer* C to the IMD.

4. **Coffer opening:** Starting with an empty set J, for each element $f^{imd}_j \in F^{imd}$, the IMD adds j to J if $f^{imd}_j \in C$. The result is an ordered set $J = \{j_0, \ldots, j_{n-1}\}$ of feature positions in F^{imd}. Opening is considered successful if $n \geq q$ for some predetermined parameter q.

5. **Key computation:** The IMD computes $\kappa = h(f^{imd}_{i_0} \parallel f^{imd}_{i_1}, \ldots, \parallel f^{imd}_{i_n})$ for a hash function h. The IMD sends (J, m, μ) to the Programmer, where $\mu = MAC_\kappa[m]$ for a message m (whose details are unimportant here).

Attack on small hash range. OPFKA has a security weakness resulting from the use of hashing in step 2. If the IMD selects in step 4. ("Coffer opening") a feature that is in $C \bigcup F^{imd}$, but not in F^{pro}, then the Programmer cannot then compute κ, and the protocol fails. To reduce the rate of such failures, the authors intend for step 2. ("Hashing") to expand the range of feature values in C.

But application of a "hash" function H *does not* expand the possible range of feature values *for a fixed domain* D. Let $D = \{0,1\}^{12}$ be the set of possible values for a 12-bit feature $\tilde{f}$. The hash of $\tilde{f}$ is computed as $H(s, \tilde{f})$, for pre-agreed salt s (a random nonce). Let $R = \{H(s, \tilde{f})\}_{\tilde{f} \in D}$ denote the range of $H(s, \cdot)$ over D. Then it is easy to see that $|R| \leq |D| = 2^{12}$, as $H(s, \cdot)$ is a deterministic function. (In fact, given the 12-bit domain and 20-bit range of H in OPFKA, with high probability over s, $|R| < |D|$.)

Thus the vast majority of chaff values in C will be invalid feature values lying outside R. Let $\hat{R} = C \bigcap R$ denote the set of values in the coffer that are valid feature values. (Note that $F_P \subseteq \hat{R}$.) The probability that a randomly selected chaff value $f' \in \{0,1\}^{20} - F^{pro}$ lies in $\hat{R}$ is bounded above by $|R|/(2^{20} - N) < 0.004$.

By excluding invalid chaff values (those not in $\hat{R}$), an adversary can greatly constrain its search space in a brute-force attack against the key κ, as shown in Algorithm 2. (Here,

Π_n denotes the set of permutations over $\mathbb{Z}_n$ and $\pi \in \Pi_n$ is a permutation $\pi : \mathbb{Z}_n \leftrightarrow \mathbb{Z}_n$.)

Algorithm 1 Key search algorithm for OPFKA

`Inputs:` $J, m, \mu, C, \hat{R}$

`Output:` Key κ

for all $< f_0, f_1, \ldots, f_n > \in \hat{R}^n$ **do**
  for all $\pi \in \Pi_n$ **do**
    $\kappa' \leftarrow h(f_{\pi(0)} \parallel f_{\pi(1)} \ldots \parallel f_{\pi(n-1)})$;
    if $MAC_{\kappa'}[m] = \mu$ **then** output κ'; halt
    end if
  end for
end for

For example, for one proposed parameterization for OPFKA ($M = 1000$, $N = 30$, and $q = 12$) a key-strength equivalent of 120 bits is claimed in [46]—well beyond feasibility for a brute-force attack. With probability about 63%, though, there will be at most 4 valid chaff values in $\hat{R}$. In this case, assuming $n = q$, the maximum running time of Algorithm 2 will be $\binom{(4+30)}{12} \times 12! \approx 2^{58}$—equivalent to breaking a 58-bit key, and requiring vastly less effort than the claimed 120-bit strength of OPFKA. Cracking a 58-bit key is within the realm of feasibility, as shown by successful cracking of a 64-bit (RC5) key in 2002 [28].

Remark: Distinct salt values might be used in hashing for different positions. This might seem more secure, but isn't, as F^{pro} would no longer in general contain valid feature values for all positions.

Adaptive attack. For large M, such as the proposed parameter $M = 5000$, OPFKA is vulnerable also to an adaptive attack in which an adversary simulates a Programmer to extract the key κ from the IMD. Due to lack of space, our description is brief.

Adv constructs a coffer C as follows. R is partitioned into (arbitrary) equal sized (size 2^{11}) sets R_0 and R_1. A coffer C is constructed that includes R_0 and a subset $R_1' \subseteq R_1$ (to

be specified); other features in C are selected to be invalid (drawn from $\{0,1\}^{20} - R$). With high probability, F^{imd} will include $n \geq q$ feature values in R_0. Therefore the IMD will respond to transmission of C with a set of indices J for any choice of R_1'.

Now, in an initial transmission, Adv sets $R_1' = R_1$. With high probability, F^{imd} will include at least one feature value in R_1' with index j. By recursing on halves of R_1' and observing whether $j \in J$ in the IMD's response, Adv can perform a binary search and learn f_j^{imd} with $log_2|D| = 12$ transmissions. By choosing different initial partitions (R_0, R_1), Adv can learn q feature values in F^{imd} and successfully impersonate a legitimate Programmer.

Variant attacks are possible with smaller M and with parallelization to search for multiple IMD feature values simultaneously.

The attack here assumes an ability to query the IMD fairly rapidly, and arises in part because OPFKA includes no throttling or back-off mechanism. A simple countermeasure is for the IMD, after a failed key-agreement with a Programmer, to refuse connections until it collects a fresh set of IPIs. Of course, this raises the risk of denial-of-service attacks and delays due to protocol failures.

2.4 Modeling

Before diving into details on H2H, some basics on ECG and our associated statistical model are in order, as well as discussion of our operational, trust, and adversarial models.

2.4.1 ECG model

Figure 2.1 is a schematic depiction of the ECG waveform of a healthy patient. The so-called R-peak is the most prominent feature of the ECG waveform; it corresponds to the "beat" in a heartbeat. The time between two consecutive R-peaks, or the heartbeat duration, is commonly referred as the inter-pulse interval (IPI). As the figure shows, a typical ECG cycle includes other physiologically significant, named features: The P-wave, which occurs

before the R-peak, the QRS complex, which includes sharp valleys before and after the R-peak, denoted by Q and S respectively, and the T-wave, following the S valley.

The heart rhythm is governed by the parasympathetic nervous system, in which many non-linearly interacting processes give the IPI its well-studied chaotic nature [16, 85]. The ECG waveform, and parasympathetic network more generally, are influenced by both long-term trends such as circadian rhythm and short-term temperature and respiratory changes. Thus ECG waves simultaneously exhibit both long term patterns and short-term chaotic behavior.

The ECG signal is well modeled as a stochastic process. The existence of long-term patterns renders the process non-stationary, meaning that the parameters of its underlying distribution, e.g., mean and variance, fluctuate over time. We introduce transforms for H2H, however, that eliminate long-term variations, creating a residual signal that is well-approximated by a wide-sense stationary stochastic process, i.e., one whose first and second moments don't change over time. Previous work observed a strongly random element in IPI time series values [16], motivating later use of IPIs as a natural source of randomness [20].

Like H2H, previous systems also exploited the natural *synchronization* property of IPIs. Slight shifts in the time interval over which IPIs are derived don't impact IPI values, which are computed relative to R-peaks, not absolute time.

Entropy and security: We show that it is possible to extract four *high-grade* random bits per IPI from our processed ECG source, i.e., bits that have maximal entropy and are fully uncorrelated. We use this entropy measure to characterize the security of H2H formally using our main theorem, Theorem 1. An important new aspect of our work is quantification and use of the different error rates incurred by individual high-grade random bits via the Neyman-Pearson Lemma. This simple approach marks a notable advance over earlier work, which assigned the same significance to all random bits. Our improvement enables authentication with the optimal, i.e., minimum possible, false positive rate for a given false negative constraint.

We also measure the total, *low-grade* entropy of the source, randomness whose full

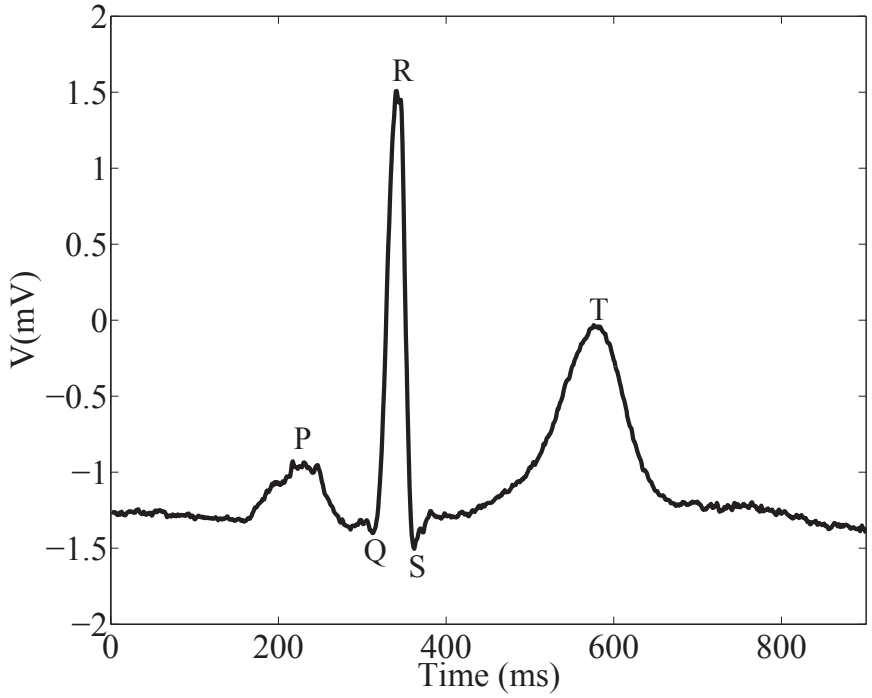


Figure 2.1: A typical ECG waveform from lead V2 which is recorded from chest. R-peak is the most prominent feature.

statistical properties in terms of correlations across bits and time are unknown. We can't use our low-grade entropy measure directly in Theorem 1. But as H2H supports arbitrary distance metrics, it permits general comparisons of ECG waves α and β. Thus we can appeal to this low-grade entropy measure as a "hedge": We believe that most of this low-grade entropy can contribute directly to the security of H2H, and thus results in a higher security level than we prove formally.

2.4.2 Operational and trust models

We envision use of H2H primarily for *emergency* authentication, when medical personnel, e.g., emergency medical technicians (EMTs), need access to a patient's IMD, but have no pre-established keys or trust relationship. Rapid and reliable access to the IMD is important. Thus H2H harvests PV randomness efficiently to achieve quick authentication.

We assume no public-key infrastructure (PKI) for certification of trustworthy programmers. The challenges of key revocation, tamper-proofing of programmers to prevent key compromise, etc., are substantial. Similarly, we assume it's impractical for medical personnel to contact an authority on the fly for access credentials, as this approach would require an infrastructure of broad (indeed, worldwide) and robust trust relationships. Thus H2H relies exclusively on the touch-to-access policy for authentication.

The ECG waveform goes flat when an acute heart attack occurs. Similarly, in some late-stage terminal diseases, the parasympathetic network collapses and as a result, the ECG waveform loses most of its entropy. The hugely distorted ECG waveform resulting from such conditions is readily identifiable. In such cases, H2H is designed to enter a promiscuous mode in which any Programmer may access the IMD: For these acute events, the risks of medical failure greatly outweigh those of malicious attack. Additionally, these extreme medical conditions occur rarely.

This limitation is prevalent across all PV-based secret extraction methods because of the underlying biological uncertainly of the variables. This security limitation however, does not significantly affect the applicability of PV-based methods since there are many

more emergency conditions that do not fully distort the ECG waveforms.

In non-emergency situations, for instance, when a patient is receiving routine medical care, it may be practical for medical personnel to retrieve device-specific keys. But in unusual situations, e.g., patients traveling abroad, lost keys, and so forth, H2H is additionally useful as a secondary or backup authentication mechanism.

2.4.3 Adversarial model

We design H2H for a strong adversarial model that assumes the presence of an attacker during a Programmer-to-IMD authentication session. This adversary is active. It has complete network control, i.e., can drop (jam), modify, replay, and forge messages at will. (The adversary can't compromise the Programmer or IMD, which would render protection of the IMD impossible.) While the presence of an adversary during a medical emergency is admittedly a strong assumption, we believe it is prudent to design robust security for critical systems such as IMDs by default.

Protecting against strong, active adversaries in a pairing protocol isn't straightforward. As mentioned above, recent breaks of two recent such protocols illustrate the challenge [100].

2.5 Authentication process

Several papers, e.g., [93, 124, 132], have proposed IMD authentication based on IPIs. Their common motif is extracting the purely uncorrelated random bits from IPIs and utilizing them as a key. As previously stated, IPIs are not independent across time. So most previous approaches were confined to utilizing only a portion of the quantized IPIs for key derivation. These protocols generate a key by quantizing IPIs and then concatenating the three or four least significant bits of the quantized IPIs. The Programmer is then authenticated if and only if the Hamming distance between the keys generated by the two communicating parties is less than a predefined threshold value.

The four least significant bits of IPIs (IPI_4) are known to be independently and identically distributed (i.i.d.). We confirmed this characteristic by studying a 2Mbit dataset consisting of 48 half-hour ECG records of 47 subjects from the MIT-BIH Arrhythmia Database [84], 549 two-minute records of 290 subjects from the PTB Database [14], and 250 records of 250 patients from MGH/MF Waveform Database [128], each roughly 30 minutes in length. All of these databases are available at [1]. We note that while these widely referenced databases contain records from patients with abnormal cardiac rhythms, these and similar databases remain standards for the study of ECG-based biometric authentication. (See, e.g., [124, 132].) Such use is substantiated by an extensive body of literature (e.g., [38,119,135]) documenting *stronger* chaotic effects in healthy hearts than in diseased ones.

We applied the NIST suite of statistical tests [114] to our dataset. The outputs of the NIST statistical tests are p-values listed in Table 2.1. These p-values represent the probability that the dataset was generated by a random process. If this value is less than a threshold (usually 1%), the randomness hypothesis is rejected. Table 2.1 shows that the p-values are all greater than 1%.

Table 2.1: p-value of several NIST statistical tests for IPI_4. These bits pass all of the random tests.

NIST test	p-value
Runs	0.311310
Rank	0.879647
Longest runs	0.185359
Frequency	0.011830
Universal test	0.013223
Approximate entropy test	0.464725
FFT test	0.131301
Linear complexity	0.612269

The lack of correlation between bits allowed previous work to use a simple Hamming distance metric to compare received and measured bits. Prior work, however, did not characterize and optimize for the *error rates* on sampled bits resulting from noisy IPI readings. More precisely, we define the error rate as the probability, for a given IPI-derived bit, that two devices, e.g., a Programmer and IMD, read the same IPI at different points on the body but output differing bit values.

Lacking the ability to obtain IPI measurements from IMDs in our lab, we estimate the error rates for IPI-derived bits in the H2H setting by means of two external ECG leads, on the left arm and right arm of subjects. The electric potential of the ECG lead III between the left arm of subjects and their left foot is taken as a surrogate for the ECG from the IMD. Similarly, the electric potential of the ECG lead II between the right arm and left foot is taken as a surrogate for the ECG recorded by the Programmer and IMD.

It should be noted that we are using the interval between consecutive prominent R-peaks, instead of analyzing the whole waveform as in [126]. Therefore, H2H is not sensitive to the location of leads on the body. Any other lead configuration could have been used in our analysis. We performed this analysis with other lead configurations and didn't find any significant variations in the error rate of least significant bits.

In the next step of analysis, the IPI values of these two readings from leads II and III are calculated and quantized. They are then converted to a Gray-code representation to minimize the difference between the quantized bits caused by error in measurement.

The Hamming distance between these two sets of bits is then taken as a surrogate for the error rate between IPI measurements by the IMD and Programmer. The results of our experiment are reflected in the "Error rate" column of Table 2.2 for IPI values quantized to 8-bit representation. The last column shows the 95% confidence interval of the error rates. We again used an aggregate of MIT-BIH [84], PTB Database [14], and MGH/MF Database [128] to estimate these error rates.

The table shows that the error rate varies considerably across quantized bits; the lower the significance of the bit, the higher its error rate and entropy. In the next subsection,

we describe our statistical approach—a departure from the naïve Hamming approach of previous work—to compare PV readings during authentication with suitable weighting for individual bit errors.

Bit	Entropy	Error rate	95% CI	Denoted by
8 (MSB)	0.27	0.001	–	–
7	0.80	0.003	–	–
6	0.90	0.004	–	–
5	0.98	0.006	–	–
4	1	0.009 (e_4)	0.008 – 0.012	x_4
3	1	0.018 (e_3)	0.015 – 0.021	x_3
2	1	0.039 (e_2)	0.035 – 0.043	x_2
1 (LSB)	1	0.080 (e_1)	0.075 – 0.086	x_1

Table 2.2: Average entropy and the estimated error rate of quantized bits, along with their 95% Confidence Interval (CI).

2.5.1 Quantifying the probability of skin contact

It's convenient to treat the IMD PV α as *correct*. Error rates then characterize honest or attacker deviation from α.

A PV in H2H includes only the four least significant bits of an IPI, which we denote collectively by IPI_4. The bits in IPI_4 are i.i.d. random variables. Thus, an adversary that hasn't made skin contact with a victim, and has no information about IPIs, can at best guess an IPI_4 value by assigning random values to each of its constituent bits. Suppose that n is the number of distinct IPI_4 instances read in an H2H authentication session. Then the total number of incorrect guesses by Adv for any given one of the four IPI_4 bit positions can be modeled as a binomial distribution with Bernoulli trial probability of 0.5, denoted by $B(n, 0.5)$.

The total number of incorrect bit outputs for a given bit position i by a valid Programmer with skin contact can be modeled by another binomial distribution $B(n, e_{\mathrm{i}})$. Here, e_{i} is the error rate of bit $i \in \{1, 2, 3, 4\}$ as given in the third column of Table 2.2.

Figure 2.4 compares the distributions of incorrect guesses by an adversary (with no skin contact) against those of a valid Programmer, for $n = 20$. The solid line is the distribution for the adversary on any of the four bit positions. For the Programmer, $x_i = B(n, e_{\mathrm{i}})$ denotes the random variable corresponding to total incorrect values in bit position i. The adversary is seen to produce significantly more errors than the Programmer in all bit positions.

2.5.2 Neyman-Pearson hypothesis testing

Recall that the goal in the H2H authentication process is to determine whether $\alpha \approx \beta$, where the IMD reads PV α and the Programmer submits PV β. Determining whether Programmer PV β is authentic, i.e., resulting from skin contact, may be viewed as a *hypothesis test*. The underlying hypothesis is that the Programmer's claimed PV β is drawn from the probability distribution of an honest Programmer, instead of an adversary's guessing distribution.

This observation motivates use of the well known Neyman-Pearson Lemma [86] to distinguish between honest and adversarial authentication attempts. Let *error value* u denote the set of errors in β, i.e., bit positions that differ from α. In our context, then, Neyman-Pearson Lemma states that for a given maximum acceptable false negative rate, the false positive rate is minimized as follows. For a fixed threshold value Th (whose computation we discuss below), a submitted Programmer value β is accepted as valid only when the following criterion holds:

$$\log\left(\frac{P(u)}{Q(u)}\right) > Th, \tag{2.1}$$

where $P(\cdot)$ denotes the probability of an adversary with no skin contact yielding error value u and $Q(\cdot)$ denotes the probability of a valid Programmer yielding u. We model

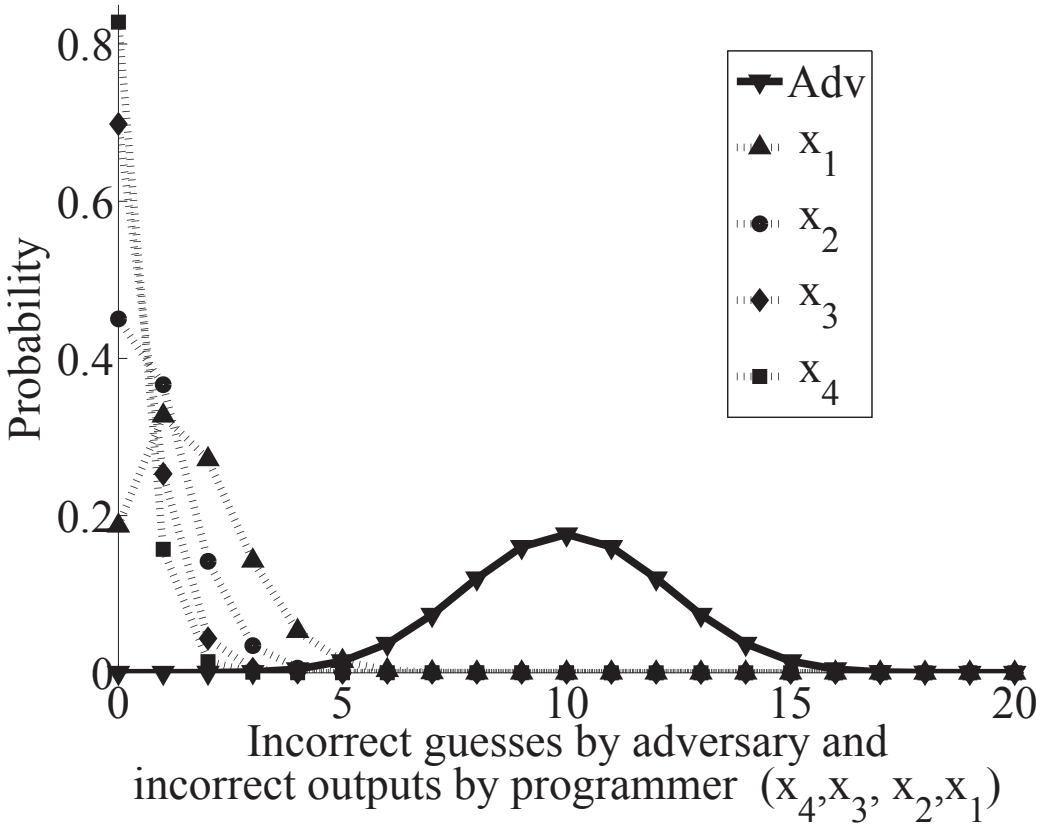


Figure 2.2: Probability distributions on incorrect guesses by a valid Programmer (dotted lines) and an adversary (solid line), for $n = 20$ (reading of 20 IPI_4 values). Here, x_1, x_2, x_3, and x_4 denote random variables on Programmer errors for bits positions 1, 2, 3, and 4 respectively. The distribution for the adversary is identical across bit positions. Clear separation is seen between valid and adversarial distributions, even for bit positions with relatively high error rates (e.g., for x_1).

distributions $P(\cdot)$ and $Q(\cdot)$ according to the binomial distributions discussed above and depicted in Figure 2.4.

We observe that as the bits in a given bit position i are i.i.d., the correctness of a PV is invariant to *which* IPI_4 values contain erroneous bits. The authenticity of a PV β is thus determined based on only the *total* number of correct or incorrect values in each bit position. So we can treat u as an *equivalence class* of PVs. In particular, it's convenient to regard u as a vector $\vec{u} = \langle u_1, u_2, u_3, u_4 \rangle$, where u_i denotes the total number of IPIs in β

that are incorrect in bit position i—again, that differ from those in α.

Now, $P(\vec{u}) = \prod_{i=1}^{4} P_i(u_i)$ and $Q(\vec{u}) = \prod_{i=1}^{4} Q_i(u_i)$, where $P_i(u_i)$ and $Q_i(u_i)$ denote the probability of a total of u_i incorrect IPI values for bit position i in adversarial and honest scenarios, respectively. It follows that:

$$\log\left(\frac{P(\vec{u})}{Q(\vec{u})}\right) = \sum_{i=1}^{4} \log\left(P_i(u_i)\right) - \sum_{i=1}^{4} \log\left(Q_i(u_i)\right). \tag{2.2}$$

Based on Equation 2.2, we can construct complete, compact representations of $P(\vec{u})$ and $Q(\vec{u})$. For a given value of n, it suffices to build a table containing $\log P_i(u_i)$ and $\log Q_i(u_i)$ for $i \in \{1,2,3,4\}$ and $u_i \in \mathbb{Z}_n$. Because the error rate of the adversary is 1/2 for *all* bit positions, $Q_i(u_i) = Q_j(u_j)$ for any $i, j \in \{1,2,3,4\}$. Thus, it suffices to store $\log Q_i(u_i)$ values for $i = 1$ only. Consequently, the full table contains just $(4{+}1){\times}(n{+}1) = 5n + 5$ values.

Given such a table, performing the Neyman-Pearson test in Equation 2.1 requires just $(4{+}1) = 5$ table lookups, eight additions, and one subtraction. This computation is *online*, i.e., performed during authentication. The storage and computational efficiency of our table-driven approach to Neyman-Pearson testing proves valuable in our implementation of H2H, described later.

One issue remains. The Neyman-Pearson Lemma states the existence of threshold Th, but doesn't specify how to compute Th. We now describe an algorithm to compute Th in our setting. Note that computation of Th takes place *offline*: Th need only be computed once and can then be programmed into an H2H-enabled IMD.

Computing Neyman-Pearson threshold value Th: It turns out that the space $(\mathbb{Z}_n)^4$ of possible values of $\vec{u}$ is relatively small. As we show below, $n \leq 50$ is sufficient to achieve our desired strength of authentication for H2H, meaning that the total number of possible values of $\vec{u}$ is at most $50^4 = 6,250,000$. Consequently, we can compute Th essentially by means of a brute force algorithm.

We specify this algorithm in pseudocode below as Algorithm 2. Algorithm 2 computes

Th for a target false-negative rate FN_{Req}. It first constructs a matrix $M[n^4][3]$ with n^4 rows, one for each $\vec{u} \in (\mathbb{Z}_n)^4$, and three columns. For each row $\vec{u}$, Column 1 contains $P(\vec{u})$, Column 2 contains $Q(\vec{u})$ and Column 3 contains $\log \frac{P(\vec{u})}{Q(\vec{u})}$.

The rows of M are sorted in ascending order with respect to Column 3 values. Then, from top (smallest) to bottom (largest), Column 1 values are accumulated in a variable p until the lowest row τ is reached for which the cumulative value $p \leq FN_{\text{Req}}$. The Column 3 value of row τ, namely $M[\tau][3]$, is the optimum threshold value Th. By summing Column 2 values over the first τ rows, we also obtain the corresponding false-positive rate (FP). (A computation failure outputs special symbol $\perp$.)

The dominant cost of Algorithm 2 is sorting. Thus its asymptotic complexity is $O(n^4 \log n)$. In practice, as n is small, the algorithm executes quickly. For example, we implemented Algorithm 2 in MATLAB on a machine with a 3.4GHz Intel i7-2600 CPU running Windows 7. It took 0.2 second to calculate Th for $n = 15$ and around 8 seconds for all values of n from 1 to 25.

Again, we emphasize that Algorithm 2 is run as a pre-computation offline, not in the IMD.

Setting parameters in H2H: In our H2H implementation, we set the false negative rate (FN_{Req}) to 10^{-4}. In practical terms, this means that a valid Programmer with skin access would fail on average in one in every 10,000 attempts; it would fail twice consecutively at most once in every 100,000,000 attempts. We believe this choice achieves adequate failure resilience for real-world scenarios.

Figure 2.3 illustrates the tradeoffs between false negative rates (FN_{Req}) and false positive rates (FP), for varying numbers n of IPI_4 values used in authentication. Table 2.3 gives detailed FP values for our implementation choice $FN_{\text{Req}} = 10^{-4}$ and, for comparison, for $FN_{\text{Req}} = 10^{-3}$. Naturally, the lower FN_{Req}, the higher FP.

In addition to FN_{Req}, the other key parameter choice in H2H is the number n of IPI_4 values measured for authentication. The larger n is, the better FN_{Req} and FP are. As n grows, though, so does the ECG measurement time in an H2H authentication.

Algorithm 2 Neyman-Pearson threshold Th computation

Inputs: n, $\{e_i\}_{i=1}^4$, FN_{Req}

Outputs: Th, FP

$P[1:n+1] \leftarrow \text{binomial}(n, 0.5)$;

for $i = 1$ to 4 **do**

$Q[1:n+1][i] \leftarrow \text{binomial}(n, e_{\text{i}})$

end for

j = 1;

for $\vec{u} = \langle u_1, u_2, u_3, u_4 \rangle \in (\mathbb{Z}_n)^4$ **do**

$M[j][1] = \prod_{i=1}^{4} P[u_i]$;

$M[j][2] = \prod_{i=1}^{4} Q[u_i][i]$;

$M[j][3] = \log(\frac{M[j][1]}{M[j][2]})$;

$j \leftarrow j + 1$;

end for

sort M on $M[\cdot][3]$ (Column 3);

$p \leftarrow 0; j \leftarrow 0$

while $p \leq FN_{\text{Req}}$ **do**

$j \leftarrow j + 1$;

$p \leftarrow p + M[j][2]$;

end while

$\tau \leftarrow j - 1$;

if $\tau < 1$ **then** output $\perp$; halt

end if

$FP \leftarrow \sum_{k=1}^{\tau} M[k][1]$;

$Th \leftarrow M[\tau][3]$;

In our implementation, we have chosen to set $n = 15$. Given our choice of $FN_{\text{Req}} = 10^{-4}$, the corresponding false positive rate is $FP = 2.7 \times 10^{-9}$. We chose this FP to demonstrate the feasibility of a *strong* level of authentication. As a point of comparison, this FP is lower than the false acceptance rate of a typical, eight-digit RSA SecurID token [104]. (While the false acceptance rate for such a token is nominally 1×10^{-8}, allowances for synchronization errors and multiple tries make it somewhat weaker.) Lower FPs, and thus lower values of n, are likely to be acceptable in practice.

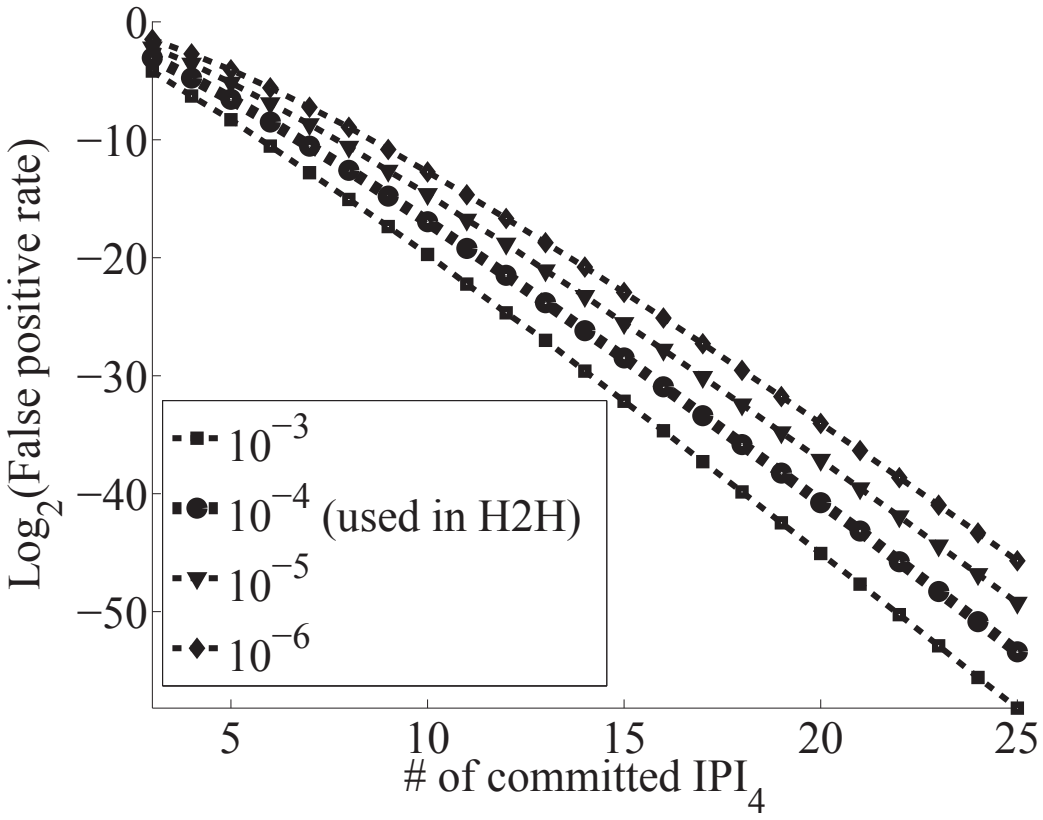


Figure 2.3: False positive rates achieved by Neyman-Pearson hypothesis testing for various false negative rates.

The average resting heart rate is 60-80 beats per minute [3]. Thus, for $n = 15$, the recording time would on average take between 7 to 10 seconds. The last column of Table 2.3 lists the average range of PV read times (in seconds) for different choices of n.

Summary: In H2H, we apply Neyman-Pearson hypothesis testing to determine whether to

Table 2.3: FP values achieved for our implementation choice $FN_{\text{Req}} = 10^{-4}$ and, for comparison, for $FN_{\text{Req}} = 10^{-3}$. Average PV read time is given in the last column.

n	$FN_{\text{Req}} = 10^{-3}$	$FN_{\text{Req}} = 10^{-4}$	Avg. read time (secs.)
5	3×10^{-3}	1.1×10^{-3}	3– 5
10	1.15×10^{-6}	7.9×10^{-6}	7–10
15	0.2×10^{-9}	2.7×10^{-9}	11–15
20	0.27×10^{-12}	5.38×10^{-13}	15–20
25	0.32×10^{-17}	8.4×10^{-17}	18–25

accept a Programmer-submitted PV β as authentic or reject it. We first compute the error value $\vec{u}$ of β. Error value $\vec{u}$ captures the total number of errors u_i in each IPI_4 bit position i in β (by comparison with the IMD PV α). We then perform Neyman-Pearson testing of $\vec{u}$ via Equation 2.1. This test involves computation using Equation 2.2, and is made efficient by precomputing a small table of u_i value probabilities and Neyman-Pearson threshold Th, both of which are stored in the IMD.

An attacker can, of course, make multiple attempts against the IMD and the Programmer as well. In Section 2.6, we formally characterize the success probability of such attacks relative to FP. Exponential backoff in the IMD is one helpful countermeasure.

2.5.3 Remote attack

We also briefly investigated an attack on H2H based on remote cardiac activity monitoring. The best reported remote heart rate monitoring result is achieved by photoplethysmography (PPG) [92]. PPG traces changes in skin color caused by temporal variations in the concentration of blood on the skin surface.

Poh et al. [92] have reported moderately accurate IPI estimation using a commercial webcam at an approximate distance of 50cm from human subjects. We reproduced and evaluated their scheme at the same distance with a 30 frame per second (FPS) camera

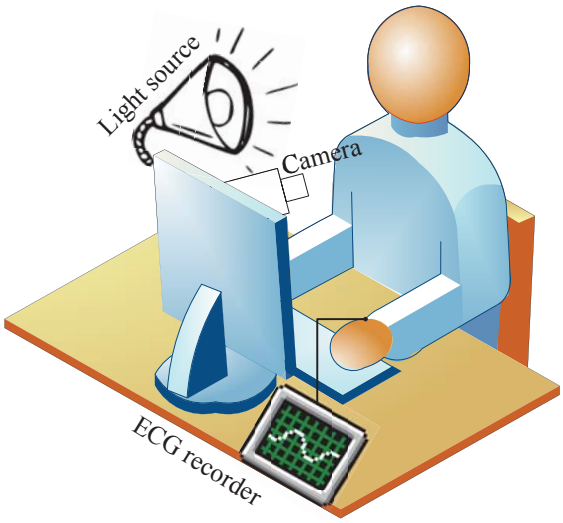


Figure 2.4: A sketch of our analysis of a remote video attack on the H2H protocol.

recording video of the subjects' faces for PPG evaluation over a two minute period. Our camera had twice the FPS rate of the camera in [92].

We have not been able, however, to achieve error rates as low as those reported in [92] for any of our test subjects. (We have contacted the authors and requested clarification of details and their original dataset for validation, but have received no response.) The average error rate for the four least significant bits (IPI4) of all four subjects in our PPG experiment were close to 50%, i.e., to random guessing— substantially higher than those achievable by a Programmer with skin access and yielding little advantage to a remote attacker targeting H2H. We conclude that without significant advances, PPG is unlikely to pose a significant threat to H2H.

A new curious remote attack against ECG-based security protocols have been recently proposed [4]. They claim that when two individuals are in close proximity, the ECG of one person gets coupled with the electroencephalogram (EEG) of the other person. This observation can be allegedly exploited by an attacker with a concealed EEG monitor to steal the ECG timings of patients by shaking hands with them. With this strategy they reported that could break the PSKA protocol 30% of times. They didnt do any proximity attacks per se, in their experiments, the subjects actually shake hand. It would be very interesting to

know how much information one can get from just standing by a person very closely. This experiment have not been done yet.

2.6 Programmer-to-IMD Pairing Protocol

We now describe the design of the H2H PV-based cryptographic pairing protocol. The IMD is a highly constrained device in terms of computational power, but particularly in terms of stored energy (battery life). A key design objective of H2H is to minimize the IMD's cryptographic computational load. First we explain our design principles, both why we don't use existing cryptographic protocols and how we exploit special features of the H2H setting to achieve simple, efficient protocol design. We then present the protocol and a security analysis.

A naïve approach to authentication might work as follows. The Programmer establishes a secure connection with the IMD (via TLS, for instance). The two devices then take respective PV readings α (IMD) and β (Programmer). The Programmer transmits β to the IMD. If $\beta \approx \alpha$, i.e., β is close to α, the IMD accepts the Programmer as valid.

The problem with this approach is that it's vulnerable to a man-in-the-middle attack. An adversary Adv can simultaneously pose as the IMD in a session with the Programmer and as the Programmer with the IMD. On receiving β from the Programmer, Adv forwards it to the IMD, resulting in a successful authentication.

Password-authenticated key-exchange (PAKE) schemes [11], are designed precisely to address such attacks, and might seem an appropriate tool for H2H. The PVs α and β measured respectively by the IMD and Programmer may be treated as passwords: The Programmer gains access to the IMD by demonstrating its approximate knowledge of "password" α, i.e., that it knows β such that $\alpha \approx \beta$.

There are two problems with PAKEs. First, due to read errors in our setting, the IMD must check for *approximate* equality, i.e., $\alpha \approx \beta$. But a PAKE requires *exact* equality. More involved approaches, e.g., bit-by-bit password testing, or use of fuzzy extraction,

e.g., [30] can convert PAKE into a "fuzzy" tool for approximate equality testing.

But PAKE presents a second problem: Computational cost. While the several modular exponentiations required by a single PAKE execution are feasible on many devices, they constitute more computation—and more energy expenditure, in particular—than desired on IMDs, which are highly constrained in terms of power and computational resources. A "fuzzy" PAKE would require even more computation.

Thankfully, as it turns out, PAKE is overengineered for H2H. It's possible to support approximate matching of α and β *and* gain better computational efficiency than PAKE.

2.6.1 Protocol overview

Our key observation is that the readings α and β in H2H are *one-time values*. In contrast to passwords, which are generally multi-use, α and β are transient. Fresh readings may be used to authenticate every session and, as we have demonstrated experimentally above, readings are statistically independent across time.

Consequently, it is possible to reveal α and β safely at the end of our authentication protocol—something not possible, of course, with static passwords. The protocol can thus rely primarily on (very fast) symmetric-key commitment and decommitment rounds and explicit IMD testing of the condition $\alpha \approx \beta$, rather than minimal-knowledge cryptographic comparison.

Our protocol has two phases: (1) A *secure-channel setup* phase, which uses (lightweight) public-key cryptography to create a secure but unauthenticated channel between the IMD and Programmer and (2) An *authentication* phase, in which the two devices use a commitment / decommitment scheme to check whether $\alpha \approx \beta$.

Secure-channel setup In the first phase of our protocol, the IMD and Programmer establish a *secure channel* via TLS. The IMD assumes the role of the TLS client; the Programmer, that of a TLS server. That is, only the Programmer presents a certificate. When instantiated with RSA, TLS requires little client computation, just one low-exponent ($e = 2^{16}+1$)

modular exponentiation.

Our protocol makes use of an output from TLS session what we call a *label* s. Given that at least one of the two entities is honest, s should be random and unique (with overwhelming probability). It isnt secret, however. In practice, s might be, e.g., the hash of the TLS master key with the public key. For convenience, we abstract away the details of TLS and just model it as a protocol SecChannel that establishes a secure channel between two entities and outputs random label s.

SecChannel (in practice, TLS) creates a secure channel in the sense that it provides confidentiality, integrity, and freshness. But it doesn't provide authentication: The IMD doesn't present a certificate, and doesn't validate the Programmer's. (As explained above, H2H avoids the burden of a PKI.) Put another way, when an IMD first sets up a secure channel, it has no assurance that it has paired with a *valid* Programmer, i.e., one actually in contact with the patient. Similarly, a Programmer doesn't know if it's communicating with a valid IMD. Thus the next protocol phase.

Authentication In the authentication phase, the two devices commit to their respective PV readings α and β. Each device binds its commitments to the label s of the secure channel on which it is communicating (preventing its re-use, prior to decommitment, on a different channel).

The IMD can then safely decommit α for the Programmer, as it has already received a commitment for β.

If the Programmer determines that $\alpha \approx \beta$, then it decommits β. Otherwise, it rejects the session. This selective decommitment helps ensure that the Programmer only reveals β to a valid IMD (one that knows $\alpha \approx \beta$), preventing re-use of β by an adversary. If the Programmer had been the party who decommits first, an adversary would have easily mounted a man-in-the-middle attack.

The IMD itself then verifies that $\alpha \approx \beta$, and makes an accept / reject authentication decision.

After an invalid authentication attempt, IMD waits a full PV read cycle before accepting a new authentication request. This delay prevents interleaving attacks, in which a Programmer's session overlaps with two IMD sessions. (This is in fact necessary to achieve Theorem 1 below.)

2.6.2 Protocol specification

The H2H authentication protocol is specified in Figure 2.5. Some technical preliminaries are needed.

Define $\mathcal{V}$ as the space of valid PVs. Let $\mathsf{dist} : \mathcal{V} \times \mathcal{V} \to \mathbb{R}_0^+$ denote a pairwise distance metric on $\mathcal{V}$. Let τ denote the time required for a device to read a PV.

We make use of a commitment scheme Commit with message space $\mathcal{V}$ and key space $\{0,1\}^k \times \{0,1\}^k$. We denote a commitment of message pair $(m, s) \in \mathcal{V} \times \{0,1\}^k$ under key $w \in \{0,1\}^k$ by $C = \mathsf{Commit}((m, s); w)$. We adopt the convention of decommitment as verification of correct commitment, i.e., decommitting m under key w involves the check $C \stackrel{?}{=} \mathsf{Commit}((m, s); w)$.

For simplicity of analysis, we treat Commit as an ideal functionality [17], i.e., as unconditionally hiding and binding. When either device outputs the message `reject`, rejecting the session, it terminates communication on the session channel. Additionally, devices support only serial sessions, not concurrent ones.

2.6.3 Privacy

H2H protects patient privacy in two senses. First, the IMD doesn't release a public key (as a Programmer does), or any other static identifier. As α is random, and protocol values are random (or pseudorandom), H2H thus provides logical-layer *tracking privacy*: An adversary can't correlate distinct RF sightings of a given IMD, i.e., can't track a patient wirelessly from a distance. (For cautions about physical-layer wireless tracking, however, see [24].) Second, the randomness of α *prevents leakage of medically significant data*, e.g.,

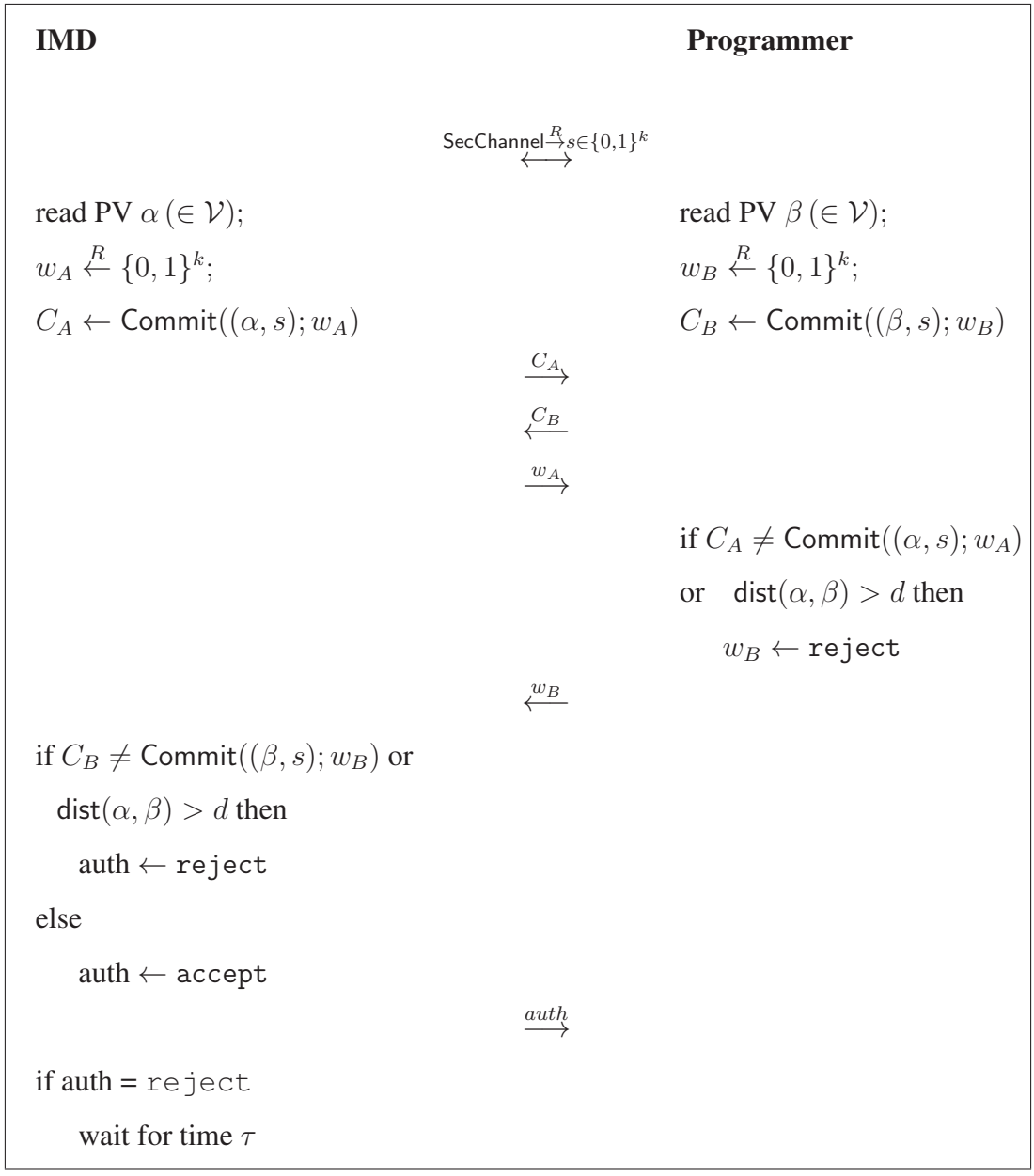


Figure 2.5: H2H pairing protocol.

cardiac abnormalities evident in a full ECG waveform.

When H2H is instrumented to use low-grade randomness, these privacy assurances are harder to substantiate. Biometric identification via ECG is possible in principle. While ECG is not as accurate as other biometrics such as fingerprints in identifying people, some success has been reported in the literature [12]. H2H, however, masks much of the ECG information exploited in such systems: It makes use only of lagged differences of the ECG waveforms, not their absolute values. When we implemented the proposed machine learning algorithm of [12] to identify the patients according to their lagged difference ECG parameters of Table 2.2, we found out that the lagged difference operation effectively removes potential privacy-infringing static information.

2.7 Security analysis

In this appendix, we briefly sketch a security analysis of the H2H authentication protocol, outlining a formal model and giving our main theorem with proof sketches. We defer a complete model, analysis, and proofs for the full version of this paper.

2.7.1 Model overview

Statistical modeling We model the process of PV sampling in terms of a set $\mathcal{V}$ of PV values and function pair $(\mathsf{sample}, \mathsf{noise})$. We call $\mathcal{M} = (\mathcal{V}; (\mathsf{sample}, \mathsf{noise}))$ the *PV model* for H2H.

At the time of PV reading a *true* PV γ is sampled from $\mathcal{V}$ under a distribution defined by probabilistic function $\mathsf{sample}(t, \tau) \rightarrow \gamma \in \mathcal{V}$, where t denotes the sampling time and τ the sampling interval length. We assume that $\mathsf{sample}(t, \tau)$ and $\mathsf{sample}(t + \tau', \tau)$ are independent and identically distributed for any $\tau' \geq \tau$. We also assume that $\mathsf{sample}(t, \tau)$ is identically distributed for any t, i.e., that it's a stationary process. Thus we let $\mathsf{sample}(\cdot, \tau)$ denote a PV sample of duration τ taken at an arbitrary time.

We model noise in PV reading by the IMD and Programmer respectively by $\alpha \leftarrow$

$\mathsf{noise}(\gamma)$ and $\beta \leftarrow \mathsf{noise}(\gamma)$, for probabilistic function $\mathsf{noise} : \mathcal{V} \rightarrow \mathcal{V}$. (A model extension can capture different noise in the IMD and Programmer.)

Cryptographic modeling We treat SecChannel as an ideal functionality. A player P can invoke SecChannel with any other player P' of its choice. The functionality then outputs a unique label $s \in \{0,1\}^k$ to P and P', or else outputs a failure symbol $\perp$. All messages labeled with s are privately delivered between P and P'; an adversary can block messages, but otherwise can't see, modify, or reorder them. Honest players support only one instance of SecChannel at a given time. Recall that we also treat Commit as an ideal functionality, i.e., perfectly hiding and binding.

Adv can at any time cause the Programmer to initate an H2H session or itself initiate an H2H session with the IMD.

Adversarial model We assume a Programmer and IMD executing serial sessions and uncorrupted by Adv. We define security with respect to an experiment involving an adversary Adv that knows $\mathcal{M}$ and fully controls the channel between the IMD and Programmer. There is a query interface send that communicates messages to the IMD and Programmer. Adv may send arbitrary queries m of the form $\mathsf{send}(entity, m)$ for $entity \in$ (IMD, Programmer). A special query $\mathsf{send}(entity, \texttt{start})$ causes a device to initiate the H2H protocol, i.e., execute SecChannel. To cause the IMD and Programmer to pair, Adv calls $\mathsf{send}(Programmer, \texttt{start})$, then sends $\mathsf{send}(IMD, \texttt{start})$) from the Programmer to IMD.

Suppose Adv sends at most q_i `start` queries to the IMD and q_r `start` queries to the Programmer over the course of the security experiment. We define $\mathsf{succ}^{H2H}_{\mathsf{Adv}}(q_i, q_r)$ as the probability that Adv causes the IMD to output `accept` for a session where it communicates with Adv on SecChannel.

2.7.2 Main theorem

We now summarize our main result. First, define:

$$p_1 = \quad max_{a' \in \mathcal{V}}(\mathsf{pr}[\mathsf{dist}(a', a) \leq d \,|\, a \leftarrow \mathsf{noise}(\gamma), \gamma \leftarrow \mathsf{sample}(\cdot, \tau)]).$$

Here, p_1 is the probability that making an *unconditioned* query, i.e., knowing $\mathcal{M}$ only, Adv can successfully guess a valid PV. (We can think of p_1 as a type of minentropy.)

Similarly, define:

$$p_2 = \quad max_{a', b' \in \mathcal{V}}(\mathsf{pr}[\mathsf{dist}(a', a) \leq d \,|\, \mathsf{dist}(b', b) > d, a \leftarrow \mathsf{noise}(\gamma), b \leftarrow \mathsf{noise}(\gamma), \gamma \leftarrow \mathsf{sample}(\cdot, \tau)]).$$

Here, p_2 is the maximum probability, given a failed PV guess b' for β, that Adv can guess a valid PV a' for a.

We have earlier presented a Lemma, which is as follows.

Lemma 1 *For PV model $\mathcal{M}$,* $\mathsf{succ}^{H2H}_{\mathsf{Adv}}(1,1) \leq p_1 + p_2 - p_1 p_2$.

We now build on this Lemma, to show that Adv maximizes its probability of success by making $q/2$ pairs of queries to the IMD and Programmer, and that its success probability for each pair of queries is at most $succ^{H2H}_{\mathsf{Adv}}(1,1)$. Theorem 1 results:

Theorem 1 *Given PV model $\mathcal{M}$ and $q = q_i + q_r$ with even-valued q,* $\mathsf{succ}^{H2H}_{\mathsf{Adv}}(q_i, q_r) \leq 1 - (1 - (p_1 + p_2 - p_1 p_2))^{q/2}$.

Proof: [sketch] Given the aforementioned Lemma, it suffices to show that Adv maximizes its probability of success by making $q/2$ pairs of queries to the IMD and Programmer, and that its success probability for each pair of queries is at most $succ^{H2H}_{\mathsf{Adv}}(1,1)$.

Given output `reject`, the IMD waits a full cycle (time τ) before initiating another session (taking input `start`). Suppose, then, that the IMD initiates local session i at time t, and thus reads $\alpha_i \leftarrow \mathsf{noise}(\mathsf{sample}(t, \tau))$. Then the IMD will only initiate a fresh session $i+1$ at time $\geq t + \tau$.

Thus if the Programmer initiates a session with PV β, then β will be independent of α_i provided that β is read at time $t + \tau$ or later. Thus, as the Programmer only initiates a session at time $t + \tau$, any Programmer PV reading β is independent of at least one of α_i or α_{i+1}. In general, then, any PV reading by the Programmer correlates with at most one α_i.

Consequently, can make at most one conditioned query, i.e., query with information about γ, per unconditioned query. It can do so only by initiating overlapping sessions with the IMD and Programmer. Given q queries in total, Adv can create at most $q/2$ such sessions. Thus, $\mathsf{succ}^{H2H}_{\mathsf{Adv}_{q_i,q_r}} \leq 1 - (1 - \mathsf{succ}^{H2H}_{\mathsf{Adv}_{1,1}})^{q/2}$.

2.7.3 Application to H2H

H2H carries two distinctive properties of uniformity that permit a simplification of Theorem 1. In particular:

- *Uniformly random PVs:* In PV probability model $\mathcal{M}_{H2H}$ for H2H, PVs are distributed uniformly at random (but correlated). That is, $\alpha, \beta \in_U \mathcal{V}$.
- *Uniform regions of validity:* For our Neyman-Pearson-derived distance metric dist_{H2H} in H2H, the number of valid PV guesses β is identical for any α in $\mathcal{V}$. (The distance between two PVs depends on their bit differences, not the PVs' specific bit values.)

Thus we can show:

Corollary 1 *Given PV model $\mathcal{M}_{H2H}$ and distance metric dist_{H2H}, and $q = q_i + q_r$ with even-valued q, $\mathsf{succ}^{H2H}_{\mathsf{Adv}}(q_i, q_r) \leq 1 - (1 - 2p_1)^{q/2}$.*

The proof is discussed in the following subsection.

2.7.4 Calculating the false postive rate of the system

In here, an upper bound for the maximum probability, given a failed PV guess b' for β, that Adv can guess a valid PV a' for a. This probability is annotated as $p_2^{\mathcal{M}}$. After a failed guess (b') for β, Adv is able to cross out b' and a space surrounding it from the sample space as possible candidates for β. This space is exactly the same as the authentication space if b' was β. In other words, the number of samples in this space is $N_s\dot{p}_1$, where N_s is the total number of samples in the sample space. Adversary will be then able to choose an other candidate for authentication. If the authentication space of the second guess doesn't overlap with the authentication space of the first guess, p_2 will be derived as $\frac{p_1}{1-p_1}$. This calculation was based on the assumption that the target of Adv, β and α are exactly the same. In practice, they are not the same. This means that the value derived for p_2 is an upper bound.

The upper bound for $p2$ can now be replaced in $p_1^{\mathcal{M}} + (1 - p_1^{\mathcal{M}})p_2^{\mathcal{M}}$ to get $2p_1$ as the upper bound for the probability of Adv success. It means thatAdv can create two "holes" in the sample space by carefully using its two guesses. The best strategy for the Adv is to select b' and a' as far as away from each other in the sample space to minimize the chances that their authentication space overlap. Flipping each committed bits in b' will do this task.

2.8 Prototype Implementation

In this section, we present a prototype implementation of H2H. A high-level architecture is shown in Figure 2.6. The IMD prototype consists of three boards:

1. A Leopard Gecko EFM-32 microcontroller (EFM32LG-DK3650);
2. an ECG analog A/D front end (TI ADS1298); and
3. a wireless sensor modem (TI CC430F5137).

Leopard Gecko is a 32-bit ARM Cortex-M3 processor with an attractive power-consumption profile and convenient power debugging tools. In our implementation, the

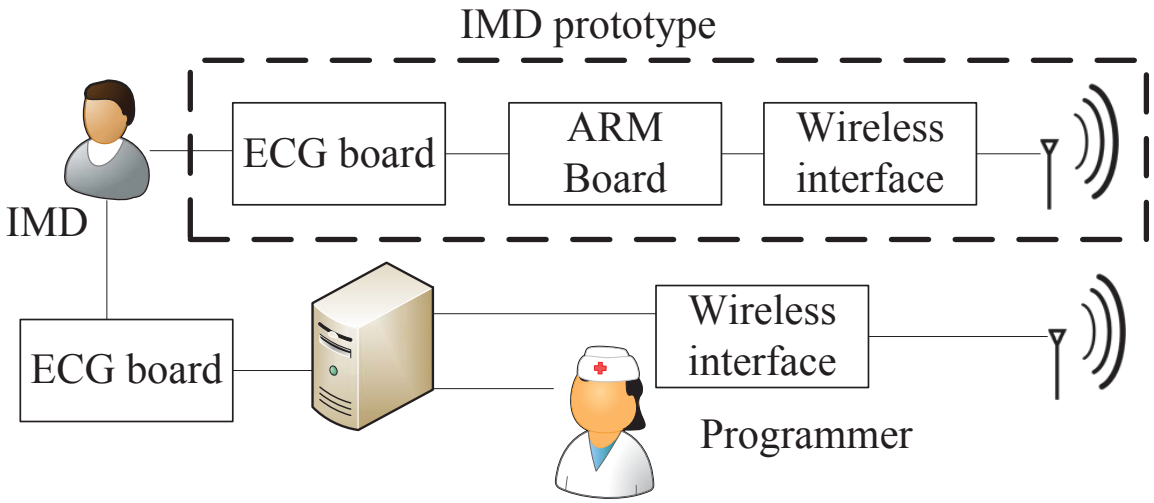


Figure 2.6: High-level view of the H2H prototype. The IMD parts are in the dotted box on top. The Programmer runs on a PC.

microcontroller communicates with the ECG analog front end and the wireless board. The EFM-32 also extracts ECG features and communicates with the Programmer using TLS. Figure 2.7 shows our implementation components. The following three subsections give details.

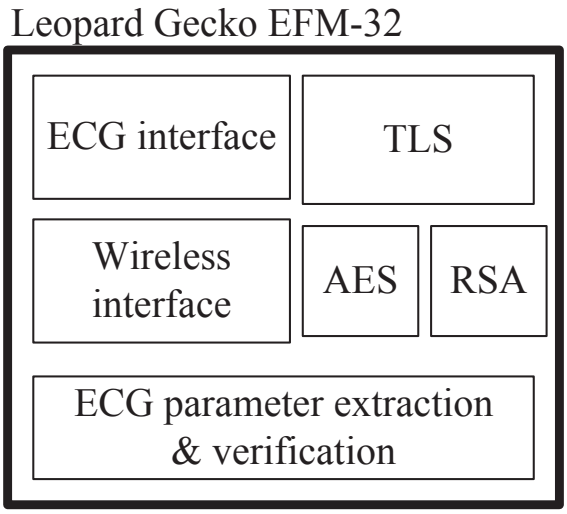


Figure 2.7: Main components of the implementation.

2.8.1 Secure channel implementation

Recall that the IMD and Programmer establish a channel between them using TLS. The IMD performs the operations of an ordinary TLS client and the Programmer those of an ordinary server.

TLS is designed to provide an encrypted and authenticated channel between two communicating parties [55]. *Standard TLS authentication assumes a PKI, however, which H2H doesn't, as noted above.* Thus the one deviation from normal TLS usage in H2H is that the IMD doesn't verify the Programmer certificate against a PKI. Instead, H2H authentication, i.e., ECG PV comparison, is performed after the TLS handshake to authenticate the channel.

Our H2H prototype uses RSA for the master secret key exchange in TLS, AES-128 for encryption, and SHA256 as the hash function. SHA256 also serves as the commitment function $\mathsf{Commit}(.)$ in the H2H pairing protocol.

We chose RSA for key exchange because RSA encryption with a small public exponent is the fastest key-exchange option for TLS [88]. In our implementation, the RSA public exponent is set to $2^{16} + 1$. The RSA modulus and message length are set to 2048 bits to conform with current NIST key length recommendations [7].

Our RSA implementation is designed to comply with the MISRA-C standard. MISRA-C [64] is a set of software development recommendations to achieve high levels of reliability for critical embedded devices. For example, MISRA-C prohibits use of dynamic memory allocations. The code size, the number of clock cycles, and approximate power consumption of various blocks of the TLS handshake are listed in Table 2.4.

2.8.2 Random number generation

H2H requires a cryptographically secure pseudo-random number generator (PRNG) for RSA ciphertext padding, key generation and nonce selection in TLS, and commitment (as shown in Figure 2.5). We use a NIST-recommended PRNG based on cipher-block chaining (CBC) [56], with AES as the underlying block cipher. The PRNG requires an initial random

Table 2.4: Approximate resource overhead of each of the component blocks of our H2H implementation. Here, "# of cycles" and "power" denote resource costs for a single call to the block.

Block name	Size (Kb)	# of cycles	Power (μ Watt)
AES encryption	2	6600	8
AES decryption	2	8400	10
RSA encryption	5	5000000	20000
MD5	1	5000	4
SHA256	3	10000	5
R-peak detection	4	5000	2

seed. We generate this seed offline and store it in the IMD's non-volatile memory. (In a commercial IMD, it can be, e.g., set at the time of manufacture.)

2.8.3 ECG parameter extraction

Our H2H prototype annotates ECG R-peaks by applying a simple length transformation to the ECG waveform using an open-source algorithm called "WQRS." In this algorithm, the arc length of the waveform over a moving window is compared against a threshold to detect heartbeats [91]. Its resource overhead is is specified in Table 2.4. (An implementation of WQRS is available on the PhysioNet website [1].)

Chapter 3

Indelible silicon randomness for lightweight secure public-key protocol

3.1 introduction

Modern security has many faces and must cover a wide spectrum of tasks. In addition to classical cryptography that provides security of stored or communicated data, modern security has to address a variety of other requirements including trust, anonymity, and privacy of actions. Earlier cryptographic methods and protocols mostly aimed to provide security for physically well-protected devices. However, a majority of contemporary devices (e.g., RFIDs or nodes in sensor networks) are easily accessible and physically unprotected, while they may even reside in hostile environments. Therefore, modern security primitives and protocols must be resilient to physical and side channel attacks. They must also be inexpensive and low power to fulfill the constraints of portable computing and communicating devices.

Classic security paradigms rely on a stored digital secret key and cryptographic algorithms. Secret keys are stored in an on-chip non-volatile memory (NVM). However, on-chip NVM storage is prone to invasive physical attacks (e.g., probing) and non-invasive imaging attacks (e.g., by scanning electron microscopes). Moreover, correct implemen-

tation of security algorithms based on a pre-distributed secret key requires Password-Authenticated Key Exchange (PAKE) protocols. These protocols are provably secure; however, they require costly exponentiation operations [11, 15, 25]. Therefore, they are not suitable for many low power resource-intensive applications.

To address these shortcomings, an alternative security approach based on the inherent, unclonable, and unique disorders of physical objects has emerged [37, 98, 101, 103, 109]. Since the physical unclonable function (PUF) is the best known security primitive in this category, we use the term PUF to generally refer to physical unclonable disorder-based security. The PUF satisfies many of the aforementioned requirements for physically securing modern and pending security applications which are not provided by classic cryptography. Furthermore, PUFs are based on elegant and intriguing concepts that are fascinating from both scientific and engineering points of view. The most popular PUFs exploit the escalating indelible process variation in modern silicon implementations of integrated circuits. Therefore, there has been a remarkable and exponentially growing interest in studying and fabricating PUFs.

Silicon PUFs use the unclonable intrinsic process variability of silicon devices to provide a unique mapping from a set of digital inputs (*challenges*) to a set of digital outputs (*responses*). The imperfections and uncertainties in the fabrication technology make cloning of a hardware circuit with the exact same device characteristics impossible, hence the term unclonable. Moreover, PUFs must be designed to make it prohibitively hard to simulate, emulate, or predict their behavior [37]. Excellent surveys of various PUF designs can be found in [2, 67, 102, 110].

Strong PUFs are a class of PUFs which have the property that the number of their possible challenge-response pairs (CRPs) has an exponential relationship with respect to the number of their physical components. This huge space of possible CRPs hinders attacks based on pre-recording and replaying previously used CRPs. However, physical components of a Strong PUF are finite. Therefore, given access to these components, a compact polynomial-order model of the CRP relationships can be built.

It should be noted that, current PUF technology has currently several significant limitations. These limitations include the limitations of secret-key based protocols which were the targets of the original PUF technology, as well as unreliability in the presence of operational or environmental variations. Additionally, it has been demonstrated that several PUF structures are easily susceptible to complete reverse engineering or at least accurate prediction of the outputs once when enough challenge-response pairs become available or side channels are exposed [69, 75, 76, 105].

A trusted IP owner with physical access to the device (e.g., the original manufacturer) can build such a compact model by measuring the PUF direct responses. Such compact models can be treated as a *secret* which can be used by a trusted Verifier to authenticate the Prover's PUF. (It should be noted that the physical access to these components should be permanently disabled before field deployment to avoid direct compact modeling.) An unfortunate fact is that third party observers may also be able to model the PUF based on a finite number of CRPs exchanged on the communication channel as it has been done before, see for e.g., [105]. This type of PUF *modeling* by untrusted third parties is also called the *machine learning* or *reverse engineering* attack as it harms the PUF security. Such attacks were possible because the challenge and response strings leak structural information about the PUF and compact models.

In this thesis, a secure, low overhead, and robust authentication and key exchange protocols for the Strong PUFs that thwart the machine learning attack. The protocols enable a Prover with physical access to the PUF to authenticate itself to a trusted Verifier. It is assumed that the trusted Verifier has access to the secret compact PUF model. The protocol leaks minimal amount of information about secret PUF parameters on the communication channel. This is because the secret is the *index* of a response substring which is randomly selected from the full response string. The Prover also adds random padding strings to the beginning and end of the substring, where the indices of the padded bits is also a part of the secret. Only the substring is sent on the channel. Since the indices are not correlated with the substring content in any ways, the secret itself is never exposed on the communication

channel. The Verifier, with access to the full string, can perform a substring matching and find the secret index. The matched strings may not be the same, but as long as they are within a small distance of each other (defined by a threshold), the matching is successful. Therefore, the method is inherently robust to the noise in the PUF responses eliminating the need for costly error correction or fuzzy extraction.

The protocol is devised such that the Verifier and the Prover jointly generate the challenges to the PUF. The challenges are generated in a way that neither a dishonest Prover nor a dishonest Verifier can solely control the challenges used for authentication. While none of the authenticating parties can solely control the challenges, the resulting challenge values are publicly known. The authentication protocol, described above, can also be leveraged to implement a low-power and secure key-exchange algorithm. The Prover only needs to select a random password and then encode it as a set of secret indices that was used in the authentication protocol.

We provide a thorough discussion of the complexity and effectiveness of attacks on proposed protocols. The protocols are designed to achieve robustness against inherent noise in PUF response bits, without costly traditional error correction modules. We demonstrate that our protocols can be implemented with a few simple modules on the Prover-side. Therefore, we do not need expensive cryptographic hashing and classic error correction techniques that have been suggested in earlier literature for achieving security. Note that recent work has used pattern matching for correcting errors while generating secret keys from a PUF [89]. However, unlike our protocol, the number of generated secret keys were limited. In addition, a higher level of protection against machine learning attacks can be achieved by our proposed protocols. To the best of our knowledge, no application of string matching for either authentication and key exchange based on Strong PUFs have been proposed before our work.

In brief, the main new contributions of our work are as follows:

- We introduce and analyze two lightweight and secure protocols based on substring-matching of PUF response strings to perform authentication and session key ex-

change. The protocols are very suitable for ultra-low power and embedded devices.

- The protocols automatically provide robustness against inherent noise in the PUF response string, without requiring externally added and costly traditional error correction modules or fuzzy extraction.
- We perform a thorough analysis of the resiliency of protocols against a host of attacks and quantify the attack complexities.
- Our analyses provide guidelines for setting the protocol parameters for a robust and low-overhead operation. In our evaluations, we derive the protocols parameter using the measurement data collected from 160 PUF instances on 10 identical FPGAs.
- The lightweight nature, security, and practicality of the new protocol are confirmed by a set of hardware implementation and evaluations.

In the following chapters, the full details of our proposed protocols are discussed. Chapter 3.2 provides a background on Strong PUFs. In Chapter 3.3, related literature is discussed and the new aspects of our work are highlighted. Authentication and key exchange protocols are described in Chapter 3.4. The parameters of our protocols and their security against multiple attacks are investigated in Chapter 3.5. The trade-offs in choosing the parameters of the protocols are explored in Chapter 3.7. Hardware implementation and performance evaluations are presented in Chapter 3.8. If the reader is familiar with PUF circuits and its related literature, he can now jump to Chapter 3.4.

3.2 Background on Strong PUFs

In this section, without loss of generality, we introduce a popular instance of Strong PUF known as arbiter PUF or delay-based PUF. Desired statistical properties of a Strong PUF are briefly reviewed, and XOR mixing of arbiter PUFs to improve the statistical properties is discussed. Note the proposed protocol can work with any Strong PUF that satisfies the requirements discussed in this section.

3.2.1 Strong PUFs and their implementation

There are a number of different PUF types, each with a set of unique properties and applications. For example, *Weak PUFs*, also known as *Physically Obfuscated Keys (POKs)* are commonly used for key generation applications. The other type is called *Strong PUF* [60]. Strong PUFs are built based on the unclonable disorder in the physical device features, with very many challenge-response pairs. The size of the CRP space is an exponential function of the number of underlying components. Strong PUFs have the property that they are prohibitively hard to clone; a complete enumeration of all their CRPs is intractable. To be secure, they should be resilient to machine learning and prediction attacks.

In this work, we use a Strong PUF implementation called "delay-based arbiter PUF" introduced in [35]. In this PUF, the delay difference between two parallel paths is compared. The paths are built identically to make their nominal delays equal by design. However, the delay of fabricated paths on chips will be different due to process variations, see Fig. 3.1. A step input simultaneously triggers the two paths. At the end of the two parallel (racing) paths, an arbiter (typically a D-Flip Flop) is used to convert the analog difference between the paths to a digital value. The arbiter output becomes one if the signal arrives at its first input earlier than the second one, otherwise, it stays at zero. The two paths are divided into several smaller sub-paths by inserting path swapping switches. Each set of inputs to the switches acts as a challenge set (denoted by C_{i}).

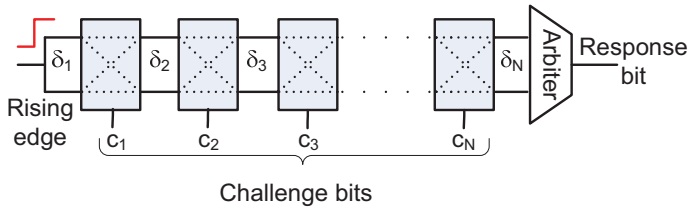


Figure 3.1: An arbiter linear PUF block with N challenges and one response bit. The arbiter converts the analog delay difference between the two paths to a digital value.

The PUF only consists of linear addition and subtraction of delay elements. Therefore,

the behavior of the PUF in Fig. 3.1 can be modeled by the following linear inequality [63]:

$$\sum_{j=1}^{N}(-1)^{\rho_j}\delta_j + \delta_{N+1} \underset{r=1}{\overset{r=0}{\lessgtr}} 0, \tag{3.1}$$

where δ_j is the differential segment delay and ρ_j is related to the input challenge that controls the switch selectors by the following relation,

$$\rho_i = \bigoplus_{x=i,i+1,...,N} C_x = C_i \oplus C_{i+1} \oplus ... \oplus C_N. \tag{3.2}$$

According to Inequality 3.1, if the difference between the sum of delays on the top and bottom paths is greater than zero, then the response will be '1'; the response is '0' otherwise. To simplify the notations, Inequality 3.1, can be rewritten as:

$$r = Sign(\Delta.\mathbf{\Phi}), \tag{3.3}$$

where $\Delta = [\delta_1, \delta_2, ..., \delta_{N+1}]$ is the delay parameter vector, $\mathbf{\Phi} = [(-1)^{\rho_1}, (-1)^{\rho_2}, ..., (-1)^{\rho_N}, 1] = [\varphi_1, \varphi_2, ..., \varphi_{N+1}]$ is the transformed challenge vector in which $\varphi_i \in \{-1, 1\}$, '.' is the scalar product operation, r is the response bit, and $Sign$ is the sign function. We will refer to $\mathbf{C}$ as the input challenge vector. Note that the parameters Φ, ρ, and C are related to each other.

3.2.2 Linear Arbiter PUF statistical properties

In this subsection, the statistical properties of a linear arbiter PUF are reviewed. It has been demonstrated in [75] that when the delay parameters $\delta \in \Delta$ come from identical symmetric distributions with zero mean (in particular it is safe to assume that the δs are independent and identically distributed Gaussian variables, i.e., $\delta \in N(0, \sigma)$), then the following statistical properties hold for a linear arbiter PUF:

- The output response bits are equally likely over the entire space of challenges, i.e, $Prob\{r = -1\} = Prob\{r = 1\} = 0.5$. Half of the challenges map to $r = -1$ and the other half maps to $r = 1$.

- The responses to similar challenges are similar. In other words, the probability that the responses r_0 and r_1 to two input challenge vectors C_0 and C_1 are different is a monotonically increasing function of the Hamming distance between the input challenges, i.e., $Prob\{r_0 \neq r_1\}$=f(HD(C_0,C_1)). * For example, in the trivial cases HD(C_0,C_1)=0, i.e. C_0=C_1, then $Prob\{r_0 \neq r_1\}$ = 0. As the Hamming distances between the input challenge vector becomes larger, the probability of having different PUF response bits increases.

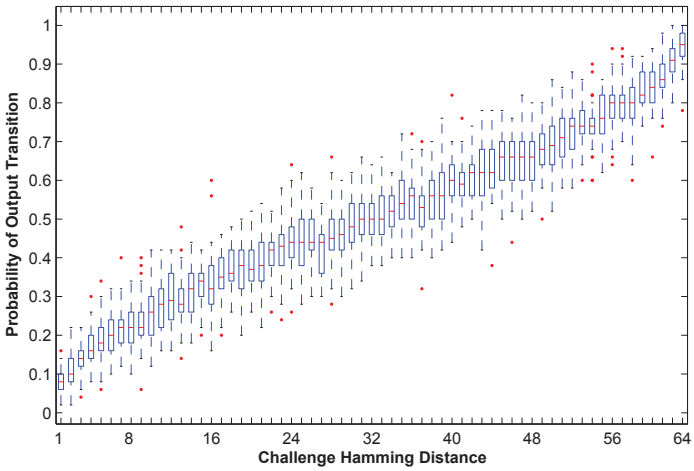


Figure 3.2: The probability of output response transition as a function of the input challenge pair Hamming distance for a population of 50 linear arbiter PUFs.

The second property leaks information about the PUF response sequence which would help in breaking the PUF security by pattern matching. Ideally, PUFs are expected to have a property called strict avalanche criterion. Any flip in the challenge bits of a PUF with avalanche criterion should cause the response bits to flip with probability of 50%. Any deviation from this criterion reduces the security of the system built based on these PUFs. To achieve this criterion, it has been proposed [75, 121] to mix the responses from the arbiter PUFs with XOR logic. In the next subsection, we review this subclass of PUFs.

*The Hamming distance between challenges C_x and C_y is defined as HD$(C_x, C_y) = \sum_{i=1}^{N} \mid C_x[i] - C_y[i] \mid /N$ where $C_x[i], C_y[i] \in \{-1, 1\}$.

3.2.3 XOR-mixed arbiter PUFs

Fig. 3.3 [75] shows a two-stage XOR-mixed arbiter PUF. In the figure, note that the challenge sequence in the second stage is applied in the reverse order. The order is flipped to help achieve the avalanche criterion. As more independent PUF response bits are mixed, the probability that output is flipped when one input bit changes, comes closer to the ideal probability of 0.5.

In addition to achieving the avalanche criterion, the XOR-mixed arbiter PUF requires a significantly larger set of challenge response pairs to successfully train the PUF model for a given target accuracy level. However, there is a cap on the number of stages that can be actually used in practice. This is due to the fact that XOR-mixing causes error accumulation of PUF responses. For instance, for a single PUF response bit error of 5%, the probability of error for a 4-XOR-mixed PUF is 19% [75]. The protocols proposed in this thesis allows higher level of security without increasing the number of XOR stages.

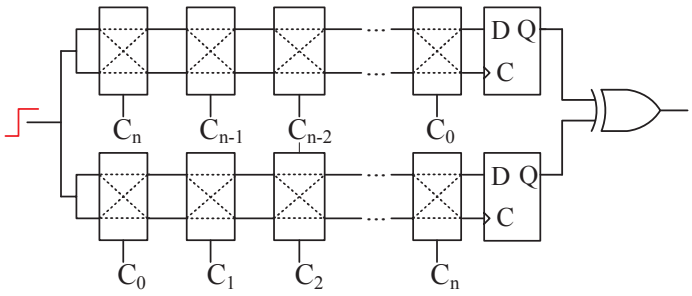


Figure 3.3: Two independent linear arbiter PUFs are XOR-mixed in order to implement an arbiter PUF with better statistical properties. The challenge sequence in the second stage is applied in the reverse order to help achieve this property.

In the following chapters, we build our protocols based on the assumption that the PUF at hand is a linear XOR-mixed arbiter PUF with near ideal statistical properties. We argue that our protocols are applicable to any Strong PUF which follows the statistical properties discussed in this Section.

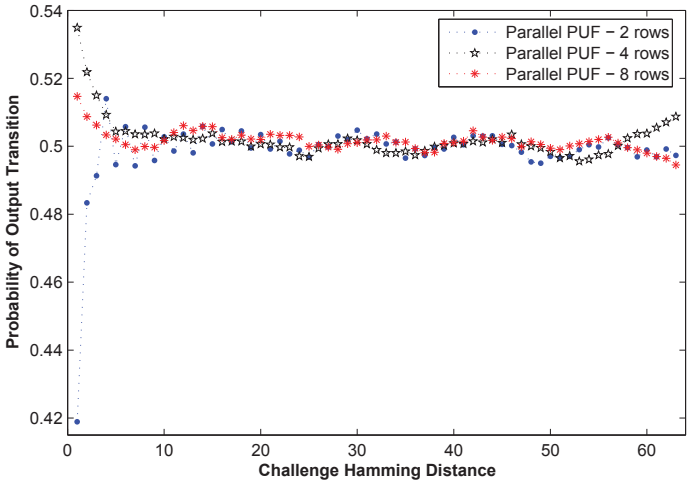


Figure 3.4: The probability of the arbiter PUF output flipping versus the Hamming distance between two challenge sequences for 2, 4, and 8 independent XOR-mixed PUFs [75]

3.3 Related work

Paradigms can be defined as generic strategies that can be easily customized to address many problems in some scientific or engineering field. For example, dynamic programming and divide-and-conquer are effective optimization paradigms. Paradigms have been essential for the development of many research domains. Shannon created two security paradigms, diffusion and confusion, in his paper [117]. The confusion paradigm states that one should use highly non-linear computational elements. Obviously, the relevant observation is that it is easy to recover the inputs from the outputs if the computational elements are linear. The diffusion paradigm insists that each output should depend on as many inputs as possible at each computational stage. Surprisingly, both confusion and diffusion are rarely explicitly used in the creation of PUF structures. We expect that they will find more use in future PUF structures and analysis.

Public key cryptography is the crown jewel of classical cryptography. Recently, several techniques have been proposed that embed PUFs in security protocols for public key communication [10, 72, 106]. These implementations are intrinsically more resilient to attacks such as power and delay analysis attacks. They also enable a very low energy realization of

public key protocols and can enable ultra fast protocols for many security protocols which are currently too slow to be practical [80, 81, 133]. There is also an ongoing effort to use PUFs to implement more complex security algorithms such as bit-commitment and oblivious transfer [107, 112]. We expect a great flurry of research activities that would further improve the initial efforts along these lines.

It is also important to consider PUF paradigms in public key security scenarios. There must be a special focus in not just creating protocols, but also ensuring that they are practical, i.e., that their energy and latency are low and their throughput is high while the required hardware is small. Public key PUF structures such as the PPUF, SIMPL, or the timed PUF also require their own paradigms. Initially, all of these methods were based on exploiting the gap between the execution time of the PUF circuitry and the required simulation time when the parameters of the pertinent PUF serve as the public key. However, in corresponding protocols, either one or both participating parties are required to conduct significant and energy intensive computational efforts, i.e. they have to either compute many challenge-response pairs or to conduct at least one long simulation. If the PUF public key has a suitable structure then the second requirement can be eliminated [133].

The first important observation is that PUFs can enable effective techniques for other security tasks such as passive and active hardware intellectual property protection, software intellectual property protection, hardware obfuscation, prevention of reverse engineering, hardware intellectual property watermarking, remote trusted sensing and computation, software metering and the prevention of hardware Trojan embedding [23, 109]. PUFs can also enable important conceptually new security tasks such self trust which enables a user to trust herself. Another important application is that PUFs can be designed in such a way that they act as synchronized hardware random number generators [81, 133] or true-random number generators [73].

The requirement for significant computation has been eliminated in second generation PPUFs. This is achieved through the use of device aging and software disabling to reduce the computational requirement to only two clock cycles. The paradigm is that the delay of

each transistor can be altered using device aging in such a way to create two identical PUFs after a subset of transistors is eliminated from the PUF using software techniques, such as special input selection [80, 81]. Much more research for implementing and evaluating this concept in current and pending technologies is still required.

Finally, digital PUFs require a completely new set of design and operational paradigms. One of them, bimodal functions, was recently proposed [133]. We expect that several conceptually new paradigms and specific techniques for digital public key PUFs will soon emerge.

Extracting secret keys from PUF responses has been explored in previous work, including [13, 36, 37, 66, 134]. Since cryptographic keys need to be stable, error correction is used for stabilizing inherently noisy PUF response bits. The classic method for stabilizing noisy PUF bits (and noisy biometrics) is error correction which is done by using helper bits or *syndrome* [29], which has a high overhead.

In the context of challenge-response based authentication for Strong PUFs, sending the syndrome bits for correcting the errors before hashing was investigated [37]; the necessity for error correction was due to hashing the responses before sending them to avoid reverse engineering. Naturally, the inputs to the hash have to be stable to have a predictable response. The proposed error correction methods in this context are classic error correction and fuzzy extraction techniques. Aside from sensitivity to PUF noise (because it satisfies the strict avalanche criterion), hashing and error correction has the drawback of high overhead in terms of area, delay, and power.

A newer information-theoretically secure Index-Based Syndrome (IBS) error correction coding for PUFs was introduced and realized in [134]. In [10], authors proposed the notion of public physically unclonable functions (PPUF) and proposed a public key-exchange protocol based on them.

All of the aforementioned methods incur a rather high overhead of error correction and/or hashing, which prohibits their usage in lightweight systems. An alternative efficient error correction method by pattern matching of responses was very recently proposed [89],

which inspired the pattern matching method used in our protocols. However, their proposed protocol and application area was limited to secret key generation.

PUFs have been subject to modeling attacks. The basis for contemporary PUF modeling attacks is collecting a set of CRPs, and then building a numerical or an algorithmic model from the collected data. For the attack to be successful, the models should be able to correctly predict the PUF response to new challenges with a high probability. Previous work on PUF modeling (reverse-engineering) used various machine learning techniques to attack both implementation and simulations of a number of different PUF families, including linear arbiter PUFs and feed-forward arbiter PUFs [36, 63, 75, 87, 105]. More comprehensive analysis and description of PUF security requirements to protect against modeling attacks were presented in [69, 74–76, 76, 105, 111]. In recent years, there have been an ongoing effort to model and protect PUFs against side channel attacks such as power analysis [69] and fault injection [26].

Finally, a new concept in PUF design has emerged that exploits nanotechnologies as the main enabler [99, 129, 130]. Recently proposed memristors and III-V nanowires are susceptible to process variations akin to traditional CMOS process. Therefore, they can be utilized to implement interesting types of PUFs. Besides process variations, memristors and nanowires also have a bidirectional and non-linear input-output behavior, which has enabled researchers to propose innovative PUF circuitry based on them [96, 129]. A very efficient authentication and key-exchange protocol based on these novel structures has been proposed in [130].

3.3.1 Rising Attacks against PUFs and PPUFs

The most nefarious attack against physical-based disorder security is cloning. For both weak and strong PUFs, cloning can be done by reverse-engineering (RE) [109]. The security of these PUFs centers on the concealment of the underlying random (unclonable) circuit variations. For weak PUFs, the adversary can learn the bounded number of secret values by RE and then clone the PUF by storing the learned values in another memory.

For strong PUFs, RE of the internal structure of the pertinent circuitry provides a basis for cloning in a compact way, without the need for storing a huge database of CRPs.

For unique objects (UNOs), where security is based on extraction of static analog fingerprints of the medium, RE can be also used for cloning purposes. As several such analog UNO structures have recently been identified [109], RE attacks on those structures is a normal topic that would follow.

There are two distinct directions for pursuing research in RE: (i) noninvasive methods, and (ii) invasive techniques. The former category can be realized in various ways, e.g., by passive/active eavesdropping of the CRPs to/from the PUF and then using a data analysis/machine learning method to model the physical PUF such as [75, 76, 105]; another noninvasive attack performs measurements on the device in operation, such as image or power *side-channels* and then integrates the measured values into the data analysis algorithm [69]. Both noninvasive methods exploit the correlations between the available information (from the side-channels or from eavesdropping) and the secret structure/values associated with PUFs.

Much more exciting research is under way in the noninvasive RE direction, including cloning the newer PUF structures, obtaining novel side-channel information, and innovations in data analysis. For example, although current machine learning methods have proven to be effective for RE of certain PUF structures, it is not immediately evident if they are directly applicable to newer structures. Furthermore, data analysis and machine learning require the art of selecting the relevant method: even if a method of choice does not help with modeling the PUF, it is mostly unclear if other methods cannot model the PUF. Therefore, investigating applicability of efficiency of innovative data analysis and machine learning methods is a timely topic.

In terms of the latter category of invasive attacks, which is often destructive, having access to the (rather) costly machinery for low-level debugging of ICs is required. As those machines are becoming more available with a higher resolution and at a lower cost, so is the susceptibility to this class of attacks.

RE, in a classic PUF sense, is not directly applicable to cloning PPUFs as the internal circuit structure of the device is assumed to be publicly available. For PPUFs, security is based upon the speed of computation of the original device; this speed should not be mimicked in simulations or emulations [94,108]. Therefore, a wicked adversary must focus on finding software or hardware structures that can find the computation results within the bounded computation time of the original device.

Perhaps because of the sparse number of physical implementations of PPUFs and the novice nature of the topic compared to (weak/strong) PUFs, a comprehensive treatment of the cloning attacks against the PPUF is yet to be done. Of particular interest would be investigating the applicability of finite state and control flow models to each class of PPUF and if applicable, defining the underlying serial and parallel computations. Naturally, parallel computations can admit a concurrent implementation; the serial computations would determine the limit of timing in terms of the number of cycles. Note that, it is our belief that the research in PPUFs is still in its infancy; alternative ways for emulating of simulating PPUF CRPs may be also explored.

It has also been shown that the power/timing consumption of PUF circuits are directly correlated with the process variation that PUF secrets are based upon. Therefore, PUFs are also shown to be susceptible to side-channel attacks [52, 68, 83, 113]. Due to effectiveness of side-channel attacks against PUFs, it is imperative that circuit countermeasures as proposed in [113] are used in future implementations. These countermeasures attempt to mitigate the correlation between the secret information and the measurable circuit delay/power consumption.

3.4 Authentication and key exchange protocols

In this section, the proposed authentication and key exchange protocols are introduced and explained in detail. The protocols are based on a Strong PUF with acceptable statistical properties, like the one shown in Fig. 3.3. The authentication protocol enables a Prover

with physical access to the PUF to authenticate itself to a Verifier, and the key exchange protocol enables the Prover and the Verifier to securely exchange secret keys between each other.

It is assumed that an honest Verifier has access to a compact secret model of the relationship between Strong PUF challenge-response pairs (CRPs). Such a model can be built by training a compact parametric model of the PUF on a set of direct challenge response pairs. As long as the PUF challenge response pairs are obtained from the linear PUF, right before the XOR-mixing stage, building and training such a compact model is possible with a relatively small set of CRPs as demonstrated in [36, 63, 75, 87, 105]. The physical access to the measurement points should be then permanently disabled before deployment, e.g., by burning irreversible fuses, so other entities cannot build the same models. Once this access point is blocked, any physical attack that involves de-packaging the chip will likely alter the shared secret.

Unlike the original PUF challenge response pair identification and authentication methodologies, our protocols are devised such that both Prover and Verifier jointly participate in producing the challenges. The joint challenge generation provides effective protection against a number of attacks. Unlike original PUF methods, an adversary cannot build a database of CRPs and use an entry in the database for authentication or key exchange. The next two subsections describe these protocols in details. The next two subsection continues the discussion by expanding the concepts of our proposed authentication protocols to implement key-exchange and bit-commitment protocols. This section will conclude by pointing out some notes about the process of secret sharing during the manufacturing process.

3.4.1 Authentication protocol steps

Fig. 3.5 illustrates the steps of our authentication protocol. Steps 1-4 of the protocol ensure joint generation of the challenges by the Prover and the Verifier. In Steps 1-2 the Prover and the Verifier each uses its own true random number generator (TRNG) unit to generate a

nonce. Note that arbiter PUFs can also be used to implement a TRNG [71]. The Prover and Verifier generated nonces are denoted by $Nonce_p$ and $Nonce_v$ respectively. The nonces are exchanged between the parties, so both entities have access to $Nonce_p$ and $Nonce_v$. Step 3 generates a random seed by concatenating the individual nonces of the Prover and the Verifier; i.e., $Seed = \{Nonce_{\text{v}} \parallel Nonce_{\text{p}}\}$.

The generated $Seed$ is used by a pseudo-random number generator (PRNG) in Step 4. Both the Prover and the Verifier have a copy of this PRNG module. The PRNG output using the seed, i.e., $C = \text{G}(Seed)$, is then applied to the PUF as a challenge set (C). Note that in this way, neither the Prover nor the Verifier has full control over the PUF challenge stream. In Step 5, the Prover applies the challenges to its physical PUF to obtain a response stream (R); i.e., $R = \text{PUF}(C)$. An honest Verifier with access to a secret compact model of the PUF (PUF_model) also estimates the PUF output stream; i.e., $R^{'} = \text{PUF\_model}(C)$.

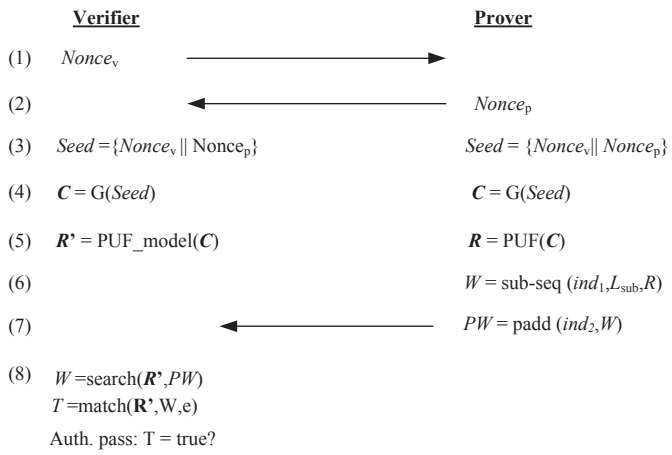


Figure 3.5: The 8 steps of PUF-based authentication protocol.

Let us assume that the full response bitstring is of length L. In Step 6, the Prover randomly chooses an index (ind_1) that points to a location in the full response bitstring. This index points to the beginning of a substring (W) with a predefined length of L_{sub}. We use the full response string in a circular manner, so if the value $(ind_1 + L_{\text{sub}}) > L$, the remainder of the substring values are taken from the beginning of the full response bitstream. This operation has been illustrated in Fig. 3.6-a.

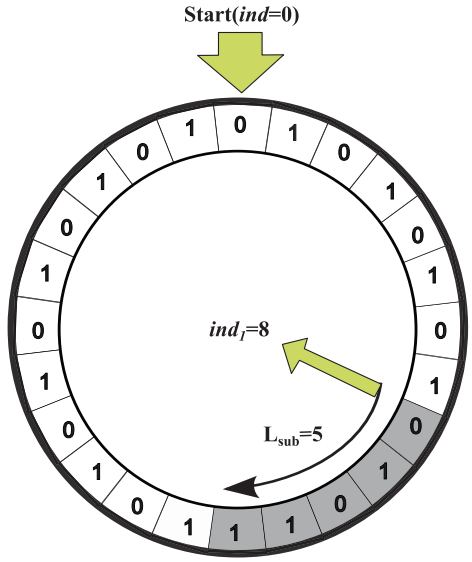
In step 7, the Prover circularly pads the substring W with random bits to create a bitstream PW of length L_{PW}. This bitstream is also referred to herein as "the padded substring". In this padding process, starting from a randomly chosen index (ind_2), the PUF substring from step 6 is inserted. We pad the substring in a circular manner. Therefore, if the value $(ind_2 + L_{\text{sub}}) > L_{\text{PW}}$, the remainder of the PUF substring is inserted at the beginning of the padded stream. This operation is illustrated in Fig. 3.6-b.

In step 8, when an honest Verifier receives the padded substring (PW), he performs a circular maximum-sequence alignment against his simulated PUF output sequence ($R^{'}$) to determine which bits belong to PUF response string and which bits are generated randomly. The authentication is successful, only if the Hamming distance between the received and the simulated substrings is lower than a predefined threshold value. After this operation, the Verifier finds out the values of the two secret indices. However these values do not affect the authentication process.

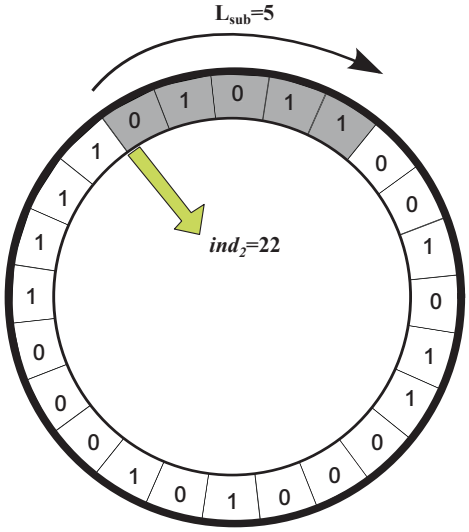
In the proposed authentication, Prover does not reveal the whole response stream and the protocol leaks a minimal amount of information. The protocol is also lightweight and suitable for ultra-low power and embedded devices. Besides a Strong PUF, the Prover only needs to implement one TRNG and one PRNG. In addition to exchanging their respective session nonces, the Prover only needs to send a relatively short padded substring to the Verifier. Additionally, the protocol has the added benefit that the ranges of the respective secret indices (ind_1, ind_2) are flexible and can be tuned depending on the security requirements. The matching threshold can also be calculated to tolerate a predefined PUF error threshold.

3.4.2 Session key exchange protocol steps

It is possible to piggyback a session key exchange protocol on the authentication protocol of Fig. 3.5. The Prover can encode secret keys as the secret indices of authentication protocol (ind_1, ind_2). The Verifier can recover these secret indices at the end of a successful authentication. If the length of secret indices is not enough to encode the whole secret key, the authentication protocol may be repeated multiple times until the required number of



(a) Circular extraction of PUF substring



(b) Circular padding of substring

Figure 3.6: The steps that are performed on the PUF response string by the Prover. Top: random selection of ind_1 and extraction of a substring with a predefined length. Bottom: circular padding the substring at a random location (ind_2) with random bits. In this toy example, $L = 24$, $L_{\text{PW}} = 24$, and $L_{\text{sub}} = 5$. Note the circular manner of extraction and padding.

secret bits is transmitted to the Verifier. We now describe this concept with an example.

If the length of PUF response string is 1024 bits, ind_1 is chosen from range of 0 to 1023. Therefore, we can encode 10 bits by using ind_1. If the length of the padded substring (L_{PW}) is 1024 bits, ind_2 is chosen from range of 0 to 1023. Therefore, 10 bits of secret key can be encoded by the ind_2. In this configuration, 20 bits overall can be exchanged between the parties with one run of the protocol. If the length of secret key is 120-bits, the protocol of Fig. 3.5 should be executed $\frac{120}{20} = 6$ times to transfer all of the secret key. This proposed protocol can securely exchange session keys with minimum overhead, while protecting against machine learning attacks and PUF response errors.

The key-exchange and authentication protocol can be followed up with a step to check whether the Verifier has received the correct indices. To do so, the Prover only needs to send the hashed values of the indices to the Verifier for verification.

3.4.3 Bit-commitment protocol

In addition to authentication and key-exchange protocols, one can easily extend the proposed concept to implement bit-commitment protocols in a similar fashion than one can implement commitment by using hashing algorithms [41].

The protocol has two phases: commitment and de-commitment. In the commitment phase, both the Prover and Verifier generate nonces. But only the Prover send the nonce to the other party. The Verifier will take the nonce of the Prover and concatenate it with his nonce and generate a pseudo-random sequence to be used as a challenge. He then uses the bits that he wants to commit as the secret indices of the protocol to send a padded substring to the Verifier.

In the de-commitment phase, the Prover reveals his secret nonce and substring index to the Verifier. The Verifier will then verifies that his nonce and bit-index can actually generate the committed substring sequence.

Like the key-exchange protocol, if the number of bits that a prover wants to commit is higher than the length of substring indices, the protocol will just needs to be repeated as

necessary.

3.4.4 Secret sharing

So far we assumed that the Verifier possesses a model of the PUF and uses the model to authenticate the Prover. The PUF in fact uses an e-fuse to protect the secret and prevent modeling attacks. The chip sets are handled by a trusted party before distributing to end users. The trusted party performs modeling on the PUF and disables the fuse before distribution. Anyone with access to the IC afterwards will not be able to model the PUF since the fuse is disabled. The trusted party can share the PUF models with other authorized trusted parties that want to authenticate the ICs.

The e-fuse mechanism is set up as follows. Before the e-fuse is disabled, the inputs to the XOR logic of arbiter PUF can be accessed from chip IO pins. This way, the Verifier can obtain as many CRPs as needed to build an accurate model of the PUF. After the model is successfully trained, the trusted party and/or the Verifier disables the e-fuse so that no one can obtain the "raw" PUF output before the XOR-mixing stage.

3.5 Analysis of attacks

In this section, we quantify the resistance of the proposed protocols against different attacks by a malicious party (Prover or Verifier). Due to similarity of authentication and key exchange protocols, similar attacks analysis apply to both of them.

In the first subsection, we quantitatively analyze their resiliency to machine learning attacks. Second, we probabilistically investigate the odds of breaking the protocols by random guessing. Third, we address the attack where a dishonest Prover (Verifier) attempts to control the PUF challenge pattern. Lastly, the effects of non-idealities of PUFs and PRNGs and their impact on protocol security are discussed. Throughout our analysis in this section, we investigate the impact of various parameters on security and reliability of protocol operation. Table 3.1 lists these parameters.

Table 3.1: List of parameters

Parameter	Description
L_{n}	Length of nonce
L	Length of PUF response string
L_{sub}	Length of PUF response substring
L_{PW}	Length of padded substring
ind_1	Index to the beginning of substring, $0 \leq ind_1 < L$
ind_2	Index at which the PUF substring is inserted $0 \leq ind_2 < L_{\text{PW}}$
N_{min}	Minimum number CRPs needed to train the PUF model with a misclassification rate of less than ϵ
k	Number of XORed PUF outputs
N	Number of PUF switch stages
th	Matching distance threshold
ϵ	PUF modeling misclassification rate
p_{err}	Probability of error in PUF responses

3.5.1 PUF modeling attack

In order to model a linear PUF with a given level of accuracy, it is sufficient to obtain a minimum number (N_{min}) of direct challenge response pairs (CRPs) from the PUF. N_{min} depends on the PUF type and also the learning strategy. Based on theoretical considerations (dimension of the feature space, Vapnik-Chervonenkis dimension), it is suggested in [105] that the minimal number of CRPs, N_{min}, that is necessary to model a N-stage delay based linear PUF with a misclassification rate of ϵ is given by:

$$N_{min} = O(\frac{N}{\epsilon}). \tag{3.4}$$

For example, a PUF model with 90% accuracy, has a misclassification rate of ϵ = 10%. In the proposed protocol, the direct responses are not revealed and the attacker needs to correctly guess the secret indices to be able to discover L_{sub} challenge response pairs. ind_1 is a number between 0 and $L-1$ (L is the length of the original response string from which the substring is obtained), and ind_2 is a number between 0 to $L_{\text{PW}} - 1$ (L_{PW} is the length of the padded substring).

Assuming the attacker tries to randomly guess the indices, he will be faced with $L \times L_{\text{PW}}$ choices. For each *iter* choice, the attacker can build a PUF model (M_{iter}) by training it on the set of L_{sub} challenge response pairs using machine learning methods.

Now, the attacker could launch $L \times L_{\text{PW}}$ rounds of authentication with the Verifier and each time use one of his trained models instead of the actual PUF. If he correctly guesses the indices and his model is accurate enough, one of his models will pass authentication. To build an accurate model as mentioned above, the attacker needs to obtain N_{min} correct challenge response pairs. If $L_{sub} > N_{min}$, then attacker can break the system with O($L \times L_{\text{PW}}$) number of attempts. However if $L_{sub} < N_{min}$, then the attacker needs to launch N_{min}/L_{sub} rounds of authentication to obtain at least N_{min} challenge response pairs. Under this scenario, the number of hypothetical PUF models will grow exponentially. Since for each round of authentication there are $L \times L_{\text{PW}}$ models based on the choice of indices value (ind_1 and ind_2), for N_{min}/L_{sub} rounds, the number of models will be of the following order:

$$(L \times L_{\text{PW}})^{\frac{N_{min}}{L_{sub}}}. \tag{3.5}$$

From the above equation, it seems intuitive to choose small values for L_{sub} to make the exponent bigger. However, small L_{sub} increases the success rate of random guessing attacks. The implications of small L_{sub} will be discussed in more detail in the next section.

The model that the attacker is building has to be only more accurate than the specified threshold during the matching. For example, if we allow a 10% tolerance during the substring matching process, then it means that a PUF model that emulates the actual PUF responses with more than 90% accuracy will be able to pass authentication. Based

on Eq. 3.4, if we allow higher misclassification rate ϵ, then a smaller number of CRPs is needed to build an accurate enough model which passes the authentication.

To improve the security while maintaining reliable performance, N_{min} must be increased for a fixed ϵ and N. This requires a structural change to delay based PUF. In this thesis, we use the XOR PUF circuit shown in Figure 3.3 for two reasons. First, to satisfy the avalanche criterion for the PUF. Second, to increase N_{min} for a fixed ϵ. Based on the results reported in the experimental evaluation section, N_{min} is an order of magnitude larger for an XOR PUF than for a simple delay based PUF.

3.5.2 Random guessing attack

A legitimate Prover should be able to generate a padded substring of PUF responses that successfully match a substring of the Verifier's emulated response sequence. The legitimate Prover must be authenticated by an honest Verifier with a very high probability, even if the response substring contains some errors. Therefore, the protocol allows some tolerance during matching by setting a threshold on the Hamming distance of the source and target substrings.

Simultaneously, the probability of authenticating a dishonest Prover should be extremely low. These conditions can be fulfilled by carefully selecting the Hamming distance threshold (th), the substring length (L_{sub}), the total length of the padded substring (L_{PW}), and the original response string length (L) by our protocol. A dishonest Prover without access to the original PUF or its model, may resort to sending a substring of random bits. In this case, the probability of authentication by a randomly guessing attacker, denoted P_{ADV}, would be:

$$P_{\text{ADV}} = (L \cdot L_{\text{PW}}) \times \sum_{i=L_{\text{sub}}-th}^{L_{\text{sub}}} \binom{L_{\text{sub}}}{i} (\frac{1}{2})^i \cdot (\frac{1}{2})^{L_{\text{sub}}-i}, \tag{3.6}$$

where L_{sub} and th are the length of the substring and the Hamming distance threshold, respectively. Eq. 3.6 is derived with this assumption that the adversary has $L \cdot L_{\text{PW}}$ chances to match the simulated PUF response, and in each match, the probability of success is

calculated using a binomial cumulative distribution function.

For an honest Prover, the probability of being correctly authenticated, denoted by P_{Honest} is:

$$P_{\text{Honest}} = \sum_{i=L_{\text{sub}}-th}^{L_{\text{sub}}} \binom{L_{\text{sub}}}{i} (1 - p_{\text{err}})^i \cdot {p_{\text{err}}}^{L_{\text{sub}}-i}, \tag{3.7}$$

where p_{err} is the probability of an error in a response bit.

If L_{sub} is chosen to be a sufficiently large number, P_{ADV} will be close to zero and P_{Honest} will be close to one.

3.5.3 Compromising the random seed

In the protocols, the Prover and the Verifier jointly generate the random PRNG seed by concatenating the outputs of their individual nonces (generated by TRNGs); i.e., $seed = \{Nonce_{\text{v}} \parallel Nonce_{\text{p}}\}$. The stream of PRNG outputs after applying the seed is then used as the PUF challenge set. This way, neither the Prover nor the Verifier has full control over generating the PUF challenge stream.

If one of the parties can fully control the seed and challenge sequence, then the following attack scenario can happen. An adversary that poses as a Verifier can manipulate an honest Prover into revealing the secret information. If the same seed is used over and over during authentication rounds, then the generated response sequence (super-string) will always be the same. The response substrings now come from the same original response string. By collecting a large enough number of substrings and putting the pieces together, the original super-string can be reconstructed. Reconstruction will reveal L CRPs. By repeating these steps more CRPs can be revealed and the PUF can be ultimately modeled.

An imposter Prover (Verifier) may intentionally keep his/her portion of the seed constant to reduce the entropy of seed. This way, the attacker can exert more control over the random challenges applied to the PUF. We argue that if the seed length is long enough this strategy will not be successful.

This attack leaves only half of the bits in the generated $Seed$ changing. For a seed of length $2L_{\text{n}}$-bits (two concatenated nonces of length L_{n}-bits), the chance that the same nonce appears twice is $\frac{1}{2^{L_{\text{n}}}}$). For example, for $L_{\text{n}} = |Nonce_{\text{v}}| = |Nonce_{\text{p}}| = 128$, the probability of being able to fully control the seed will be negligibly small. Therefore, one could effectively guard against any kind of random seed compromise by increasing the nonce lengths. The only overhead of this approach is a twofold increase in the runtime of the TRNG.

3.5.4 Substring replay attack

A dishonest Prover may mount an attack by recording the padded substrings associated with each used $Seed$. In this attack, a malicious Prover records the response substrings sent by an honest Prover to an honest Verifier for a specific $Seed$. The recording may be performed by eavesdropping on the communication channel between the legitimate Prover and Verifier. A malicious party may even pre-record a set of response substrings to various random $Seed$s by posing as a legitimate Verifier and exchanging nonces with the authentic Prover.

After recording a sufficiently large number of $Seeds$ and their corresponding response substrings, the malicious party could attempt to impersonate an honest Prover. This may be done by repeatedly contacting the legitimate Verifier for authentication and then matching the generated $Seed$s to its pre-recorded database. This attack could only happen if the $Seed$s collide. Selecting a sufficiently long $Seed$ that cannot be controlled by one party (Subsection 3.5.2) would hinder this collision attack.

Passive eavesdropping is performed during the pre-recording phase..The chances that the whole $Seed$ collides will be $1/2^{L_{\text{n}}}$ and the worst-case scenario is when an adversary impersonates a Verifier and controls half of the seed which reduces the collision probability to $1/2^{L_{\text{n}}/2}$.

3.5.5 Exploiting non-idealities of PRNG and PUF

Thus far, we assumed that the outputs of PRNG and PUF are ideal and statistically unbiased. If this is not true, an attacker may resort to exploiting the statistical bias in a non-ideal PRNG or PUF to attack the system. Therefore, in this section we emphasize the importance of the PUF avalanche criterion for securing against this class of attacks.

If the PUF has poor statistical properties, then the attacker can predict patterns in the generated responses. The attacker can use these predicted patterns to guess a matching location for the substring. In other words, statistical bias in the responses will leak information about the values of secret indices.

Recall that an ideal Strong PUF should have the strict avalanche property [76]. This property states that if one bit of the PUF's input challenges is flipped, the PUF output response should flip with a $\frac{1}{2}$ probability. If this property holds, the PUF output for two different challenges will be uncorrelated. This probability can be almost achieved when at least more than two independent PUF output bits are mixed by an XOR. As more independent PUF response bits are mixed, the probability of a bit flip in the output due a one bit change in the input moves closer to the ideal case; however, this linearly increases the probability of error in the mixed output. For instance, for a single Strong PUF response bit error of 5%, the probability of error for 4-XOR mixing is reported to be 19% in [76].

In our implementation, Linear feedback shift registers (LFSRs) are used as a lightweight PRNG. An ideal LFSR must have the maximum length sequence property [5]. This property ensures that the autocorrelation function of the LFSR output stream is "impulsive", i.e., it is one at lag zero and is $\frac{-1}{N}$ for all other lags, where N is the LFSR sequences length. N should be a sufficiently large number, which renders the lagged autocorrelations very close to zero [5]. Therefore, if an LFSR generates a sequence of challenges to the PUF, the challenges are uncorrelated. In other words, for an ideal LFSR, it is highly unlikely that an attacker can find two challenges with a very small Hamming distance.

Even if the attacker finds two challenges with a small Hamming distance in the sequence, the output of our proposed PUF would be sufficiently uncorrelated to the Ham-

ming distance of the input challenges. Therefore, a combination of PRNG and PUF with strict avalanche criteria would make this attack highly unlikely. It is worth noting that it is not required by any means for the PRNG to be a cryptographically secure generator. The seed in the protocol is public and the only purpose of the PRNG is to generate sequences of independent random challenge vectors from the Prover and Verifier nonces.

3.5.6 Man-in-the-middle attack on key exchange

Asymmetric cryptographic algorithms, such as RSA and Diffie-Hellman, are traditionally used to exchange secret keys. These asymmetric algorithms are susceptible to man-in-the-middle attacks [88]. Therefore, a certificate authority is necessary for a secure implementation of these algorithms. However, our proposed key exchange algorithm is not susceptible to man-in-the-middle attack and no certificate authority is required for implementation.

An attacker, who intercepts the padded PUF substring, does not know the PUF response string. Therefore, he does not know the value of secret indices and he cannot change the padded PUF substring to forge a specific key. An attacker, however, can possibly rotate the padded substring to add or subtract from the secret value of ind_2. Even in this case, the attacker does not know the new value of ind_2 and cannot act upon it to open a forged encrypted channel. Rotating the padded substring will only result in a denial of service attack which is already possible by jamming.

3.6 Further hardening of PUF against machine learning attacks

The XOR-mixed arbiter PUF used in this work is an improvement upon earlier version of the XORed PUFs. Here, we can approach closer to the perfect avalanche criterion, therefore much better security levels can be achieved. Without, the proposed algorithms can be broken by brute-force machine learning attacks as proposed in [9]. Despite the

improvements, these PUFs are not still perfect. In this section, we will discuss two more approaches to harden arbiter PUFs against machine learning attacks.

3.6.1 Achieving optimum order XORed PUFs

In lightweight XORed PUFs, the order of inputs to the arbiter flips in even-numbered arbiter PUFs. For the case of 4-XORed arbiter PUFs, the order of inputs will be:

1. From 1 to n
2. From n to 1
3. From 1 to n
4. From n to 1 (*)

An interesting approach to achieve near perfect avalanche criterion, one can analyze the four PUFs, and then change the input order of one of the PUFs (hear the fourth one), in order to equalize the statistical impact of each input challenge-bit.

This step can be performed at the manufacturing step, and the reordering can be implemented by deploying additional MUXes on the chip and burning the necessary fuses. We should assume that the perfect order of inputs for each chip will now be revealed to the adversary if he analyzes the fuses by a microscope.

Despite its potential improvement upon existing schemes, this approach might not be viable since it requires an expensive optimization step for manufacturing each individual chip.

3.6.2 Changing the order of inputs on the fly

There is a better way to leverage the order of inputs to the PUFs to harden them against machine learning attacks. For XORed arbiter PUFs with more than three linear PUFs, one can change the order of inputs to one of arbiter PUFs randomly and still achieve acceptable

statistical characteristics. For the case of 4-XORed arbiter PUFs, the order of inputs will be:

1. From 1 to n
2. From n to 1
3. From 1 to n
4. Randomly chosen during the protocol (*)

In the above example, the input order of the fourth linear PUFs is generated by applying a secure PRNG to the concatenated seeds of the Prover and Verifier. Here the PUF was the number 4, but this can also be selected randomly.

In this scheme, the model of the XORed PUF is also changing on the fly, therefore learning become harder. This scheme has the advantage of not needing much additional hardware or software overhead. All needed is is continue running the PRNG and then have some MUXes in the chip to reorder the inputs according to the result of the PRNG. The number of needed muxes is in the order of $\log(n)$, where n is the number of input bits for each linear PUF.

Based on our preliminary analysis, this approach increases the required time of learning a 16-bit 4-XORed PUF by more than one order of magnitude. Combining this approach with the aforementioned protocols will further enhance the security of the systems based on PUFs.

3.7 Trade-offs in protocol parameters

In this section, the trade-offs in choosing the parameters of the protocols are explored by analyzing the PUF measurement data collected in the lab. False acceptance and false rejection probabilities depend on PUF error rates. There have been no comprehensive reports till this date on PUF response error rates (caused by variations in temperature and

V_{DD} \ Temperature	5^oC	35^oC	65^oC
0.95 V	8.4%	6.2%	7.1%
1.00 V	6.8%	3.1%	6.4%
1.05 V	7.2%	6.7%	7.9%

Table 3.2: Average bit error rate of PUF in different voltage and temperature conditions in comparison with the ideal PUF output at nominal condition.

power supply conditions) nor any solid data on modeling error rates measured on real PUF challenge response pairs. The data reported in the related literature mainly come from synthetic (emulated) PUF results rather than actual reliable PUF measurements and tests.

3.7.1 experimental set up

We used the data we measured and collected across 10 Xilinx Virtex 5 (LX110) FPGAs at 9 accurately controlled operating condition (combination of different temperatures and power supply points). Each FPGA holds 16 PUFs and each PUF is tested using 64,000 random challenges.

Ideal PUF responses are obtained by challenging the PUF 128 times at the nominal condition (temperature = 35^oC and V_{DD} = 1 V) and then taking a consensus of these responses. The error rate is now defined as the percentage deviation from the consensus response. For example if 10 bits from the 128 bits are ones and the rest are zeros, the deviation from the majority response, or the response error rate, is $(10/128)\times 100 = 7.8\%$. Table 3.2 shows the average deviation (taken over 64,000 challenge-response pairs) of these experiments from the ideal response at the nominal condition. As it can be seen from this table, the error rate is substantially higher in non-nominal conditions. The worst case scenario happens when the temperature is 5^oC and the voltage is 0.95V. The table shows that 30^oC degree change in temperature will have a bigger effect on the error rate than a 5% voltage change.

Temperature / V_{DD}	5^{o}C	35^{o}C	65^{o}C
0.95 V	13.2% (*)	10.5%	10.7%
1.00 V	8.9%	6.4%	8.9%
1.05 V	9.3%	10.2%	11.8%

Table 3.3: Average bit error rate of the Verifiers PUF model against the PUF outputs in different voltage and temperature conditions. *: the worst-case scenario.

As mentioned earlier, the Verifier repeatedly tests the PUF in the factory to obtain a consensus of the PUF responses for an array of random challenges. The Verifier then uses the reliable response bits to build a PUF Model for himself. When the PUF is deployed in the field, the Prover challenges its own PUF and send the responses to the Verifier. The average error rate of the Prover response in different working conditions against the Verifiers model is listed in Table 3.3. The listed errors are the compound of two types of error. The first type is the error in PUF output due to noise of environment as well as operating condition fluctuations. The second type is the inevitable modeling error of the Verifiers PUF model. These error rates are tangibly higher than the error rates of Table 3.2. The worst error rate is recorded at 5^{o}C temperature and voltage of 0.95V. This error rate is taken as the worst-case error rate between an honest Verifier and an honest Prover. We will use this error rate to estimate the false acceptance and false rejection probability of the authentication protocol.

3.7.2 Modeling attack complexity and protocol parameters

As explained earlier, the attack complexity depends exponentially on the minimum required number of challenge response pairs (CRPs), i.e., N_{min}, to reach a modeling error rate of less than 'th', the matching threshold in the protocol. The matching threshold in the protocol is incorporated to create a tolerance for errors in the responses caused by modeling error as

well as errors due to environment variations and noise.

By relaxing the tolerance for errors in the protocol (i.e., increasing 'th'), we basically increase the probability of attack. In contrast, by lowering the tolerance for errors, the rate at which the authentication of a genuine PUF fails due to noisy responses increases. As a rule of thumb, the tolerance has to be set greater than the maximum response error rate to achieve sensible false rejection and false acceptance probabilities.

Once the tolerance level (th) is fixed to achieve the desired false rejection and false acceptance probabilities, N_{min} must be increased to hinder modeling attacks. However,N_{min} and th are inter-related for a given PUF structure. In other words, for a given fixed PUF structure, increasing th mandates that a less accurate model can pass the authentication, and that model can be trained with a smaller number of CRPs (smaller N_{min}). The only way to achieve a higher N_{min} for a fixed th is to change the PUF structure.

Earlier, we proposed using XOR PUFs instead of a single arbiter-based PUF in order to increase N_{min} for a fixed th. As reported previously in the related literature, XOR-ing the PUF outputs makes the machine learning more difficult and requires a larger CRP set for model building. The major problem with XORing the PUF outputs is error accumulation. For example, if the outputs of two arbiter-based PUFs are mixed with XORs, the XOR PUF response error rate will be about the sum of each individual arbiter-based PUF's errors. This means the error tolerance has to be doubled to have reliable operations. This observation of trade-off between N_{min} and th, led us to quantify this effect.

In order to quantify the trade-off between N_{min} and th, we first calculate the effective compound error rate of XOR-mixed PUF outputs for different operating conditions and different numbers of PUF stages. Tables 3.4, 3.5, 3.6 show the effective response error rate for 2-input, 3-input, 4-input XOR PUF respectively.

According to the above tables, the maximum error rates measured from the XOR PUF responses are 24.7%, 34.6%, and 43.2% for 2-input, 3-input, 4-input XOR-ed PUF, respectively. To guarantee reliable authentication at all operating conditions, the error tolerance of protocol (th) must be set above the maximum error rates. Now after deriving the PUF

Temperature / V_{DD}	5^{o}C	35^{o}C	65^{o}C
0.95 V	24.7%	19.9%	20.3%
1.00 V	17.0%	12.4%	17.0%
1.05 V	17.7%	19.4%	22.2%

Table 3.4: 2-input XOR.

Temperature / V_{DD}	5^{o}C	35^{o}C	65^{o}C
0.95 V	34.6%	28.3%	28.8%
1.00 V	24.4%	18.0%	24.4%
1.05 V	25.4%	27.6%	31.4%

Table 3.5: 3-input XOR.

Temperature / V_{DD}	5^{o}C	35^{o}C	65^{o}C
0.95 V	43.2%	35.8%	36.4%
1.00 V	31.1%	23.2%	31.1%
1.05 V	32.3%	35.0%	39.6%

Table 3.6: 4-input XOR.

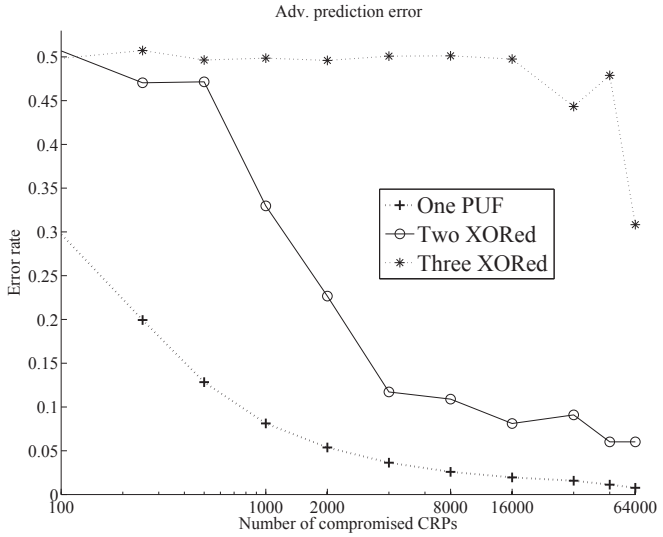


Figure 3.7: The modeling error rate for arbiter-based PUF, and XOR PUFs with 2 and 3 outputs as a function of number of train/test CRPs.

error rate, we would like to know how many challenge response pairs are required to train the PUF model and reach a modeling error rate that falls below the tolerance level. In other words, we need to know how many challenge/response pairs the adversary needs to collect in order to pass the authentication and break the system.

To answer this question, we trained and tested the PUF model on the data collected in the lab from real PUF implementations. We measured the modeling accuracy as a function of train/test set size for each PUF. The results in Figure 3.7 show the modeling error using evolutionary strategy (ES) machine learning methods.

Based on the results in Figure 3.7, the largest value of N_{min}, after taking into account the error threshold (th) derived earlier, is achieved by a 3 stages XORed-PUF. In other words, 64,000 CRPs must be collected to achieve a modeling error rate of less than 34.6%. Therefore, N_{min} = 64,000 for 3-stage XOR-ed PUF.

Table 3.7 shows the false rejection and false acceptance error rate of our protocol with the length of PUF response sequence and the length of additional pads fixed at 1028 and 512, respectively. False rejection rate is the rate in which the service to the truthful Prover is disrupted, it is calculated using Eq. 3.6: $1 - P_{\text{ADV}}$.

L_{sub}	1250		
Error threshold	487	477	467
False rejection	0.2%	1%	5%
False acceptance	9e-10	0	0

Table 3.7: False rejection and acceptance error probabilities for different protocol parameters.

The requirements on the false rejection rate are not usually as stringent as the requirements on the false acceptance rate, however, one should assume that a customer would deem a product impractical if the false rejection rate is higher than a threshold. In our protocol design, we tune the system parameter to achieve a false negative rate of 1%, while minimizing the false acceptance rate. Also, we take the worst-case error rate as the basis of our calculation of false acceptance and false rejection rates. The error rates that we report are the upper bound of what can be observed in the field by a customer/Prover.

Table 3.7 shows that the desired false rejection rate of 1% with an acceptable false acceptance rate is achieved when L_{sub} = 1250 and the error threshold is 477/1250 = 38%. In this scenario, an adversary needs to perform $O((1300 \cdot 512)^{(64000/1250)}) \approx$ O(2^{988}) machine learning attacks in order to break this system which makes the system secure against all computationally bounded adversaries.

At the end, it should be noted that the worst case bit error rate of our PUF implementation (13.2% in Table 3.3) is much higher than a recently reported bit error rate of arbiter PUFs [54] ($\approx 3 - 5\%$). The discrepancy might be explained by the fact that their implementation is based on a 65nm ASIC technology and ours is based on a Virtex 5 FPGA. Therefore, the reported security performance of our protocol has the potential to be further enhanced by a more custom implementation with a lower bit error rate.

3.8 Hardware implementation

In this section, we present an FPGA implementation of the proposed protocol for the Prover side on Xilinx Virtex 5 XC5VLX110T FPGAs. Fig. 3.9 summarizes the required resources on Prover and Verifier sides of the protocols. Since there is a stricter power consumption requirement on the lightweight Prover, we focus our evaluation on Prover implementation overhead. The computation on the Verifier side can run solely in software, however, the computation on the Verifier may also be carried out in hardware with negligible overhead.

It is desirable to use a low overhead PUF implementation, such as the one introduced in [73]. If an ASIC or analog implementation of the PUF is required, the ultra-low power architecture in [71] is suitable for this protocol. A very low-power Verifier implemented by a microcontroller such as TI MSP430 can easily challenge the PUF and run the subsequent steps of the protocol.

We use the implementation of the arbiter-based PUF in [77]. The arbiter-based PUF on FPGA is designed to have 64 input challenges. In total, 128 LUTs and one flip-flop are used to generate one bit of response. To achieve a higher throughput, multiple parallel PUFs can be implemented on the same FPGA.

There are various existing implementations for TRNGs on FPGAs [59,122]. We use the architecture presented in [73] to implement a true random number generator. One embodiment of the TRNG architecture is shown in Figure 3.8. This TRNG operates by enforcing a meta-stable state on flip-flops through a closed loop feedback system. This TRNG has a Tunable PUF as its core that consumes 128 LUTs that are packed into 16 CLBs on Virtex 5. In fact, the PUF of the TRNG is identical to the arbiter-based PUF except that the switches act as tunable programmable delay lines. The core is incorporated inside a closed-loop feedback system. The core output is attached to a 12-bit counter (using 12 registers) which monitors the arbiter's meta-stability. If the arbiter operates in a purely meta-stable fashion, the output bits become equally likely ones and zeros. The counter basically measures and monitors deviations from this condition and generates a difference feedback signal to guide the system to return back to its meta-stable state. The counter output drives an encoding

table of depth 2^{12} where Each row of encoding table contains a 128-bit word resulting in a 64KByte ROM. A table of size $2^{12} \times 8$-bits (=4KByte) implemented by a RAM block is used to gather and update statistics for online post processing.

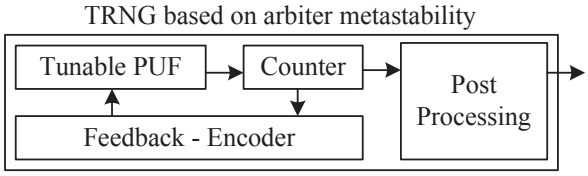


Figure 3.8: True random number generation architecture based on flip-flop meta-stability

The nonce size is set to 128 for both the Prover and Verifier. We first concatenate these two nonces and then fed the result to a 256-bit secure LFSR to generate a pseudo-random sequence.

The substring selection can be achieved by shifting the intended substring of the PUF responses into a FIFO. The shifting operation, however, begins only when needed. In other words, before running the PUF, the random indices are generated by the TRNG. For example, if the response sequence has a length of 1300 which results in a 11-bit index. To generate a 11-bit random index, we have to run the TRNG 8×11 clock cycles according to Table 3.8. Since we do not need the response bits that are generated before and after the substring window. Therefore, we do not need to even generate or store those bits and the PUF has to only be challenged for the response bits in the substring. This significantly reduces the overall run time and the storage requirement on the FIFO. The FIFO size is accordingly equal to the length of the substring.

The propagation delay through the PUF and the TRNG core is equal to 61.06ns. PUF outputs can be generated at a maximum rate of 16Mbit/sec. Post-processing on the TRNG output bits can lower the throughput from 16Mbit/sec to 2Mbit/sec. Since the TRNG is only used to generate the nonce and the indices, we can run TRNG before the start of the protocol and pre-record these values. Therefore, its throughput does not affect the overall system performance.

The implementation overhead of our proposed authentication protocol is much less than

Table 3.8: Implementation overhead on Virtex 5 FPGA

No.	Type	LUT	Registers	RAM blocks	ROM blocks	Clock Cycles
4	PUF	128	1	0	0	1
1	TRNG	128	12	4KB	64KB	8
1	FIFO	0	1250	0	0	N/A
2	LFSR	2	128	0	0	N/A
1	Control	12	9	0	0	N/A
Total		652	1400	4KB	64KB	N/A

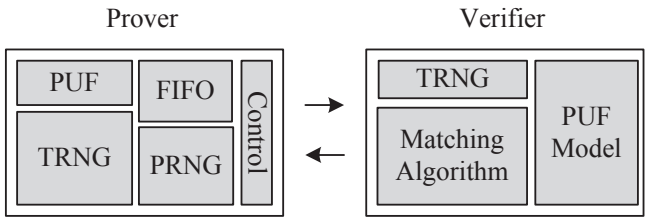


Figure 3.9: Resource usage on Prover and Verifier sides

traditional cryptographic modules. For example, robust hashing implementation of SHA-2 as implemented in [58] requires at least 1558 LUTs of a Virtex-II FPGA and it takes 490 clock cycles to evaluate. This overhead will occur on the top of the clock cycles required for PUF evaluation. Table 3.9 also has the overhead of another reported SHA-256 overhead, which is clearly higher than the overhead of our proposed approaches.

The overhead of our key exchange protocol should be compared against symmetric key-exchange algorithms not asymmetric key-exchange ones, since our protocol assumes that a secret PUF as a token has been pre-distributed between the Provers. Our key exchange protocol achieves desired level of security with minimal computational overhead. For example, AES-128 as implemented in [32] requires at least 738 LUTs of a Virtex-V FPGA, which is higher than the combined overhead of our authentication and key-exchange as listed in Table 3.8.

Table 3.9: SHA-2 implementation overhead as reported in [79]

SHA-256	Freq. (MHz)	Clock Cycles	TP (Mbps)	Area (LUTs)
Basic	133.06	68	1009	5492
2x-unrolled	73.97	28	996.7	8128
4x-unrolled	40.83	23	908.9	11592

Chapter 4

Indelible biometric randomness for secure distributed storage

4.1 The need for distributed biometric storage

The news of successful advanced persistent attacks and hacks against supposedly secure servers such LinkedIn [90] is no longer surprising. What is alarming about these hacks is the fact that they go unnoticed for a long period of time [123]. Unfortunately, governments and institutions around the world are now creating massive database of biometric markers to keep track of their citizens and employees at an alarming rate [8] and unlike the case of passwords, biometric markers are irrevocable: once lost, it is lost forever. To the best of our knowledge, no reliable and provably secure systematic approach in securing these important databases exist. The goal of this work is to address this issue. In the next section, the statistical characteristics of fingerprints are briefly discussed.

4.2 Statistical details of fingerprint randomness

High level of entropy in fingerprints is a common misconception [45, 48, 78], and the best false positive rate achievable is in order of one in a million or so. Despite this fact, finger-

prints are the biometric of choice in criminology and identification due to its convenience and public acceptance. Therefore, a reliable and secure storage protocol for fingerprint databases is long overdue.

Common password storage methods such as [50] cannot be directly applied to fingerprint databases, since the indelible randomness of fingerprint biometric is mostly comprised of fingerprint minutiae point locations and orientations and not on well-defined words or bits. Minutiae points are the locations in which the ridges of the fingerprint abruptly change or discontinue The ridge ending is the point in which a rich ends and bifurcation is a point in which a ridge is divided in two ridges. A short ridge or dot is a significantly shorter ridge on the fingerprint.

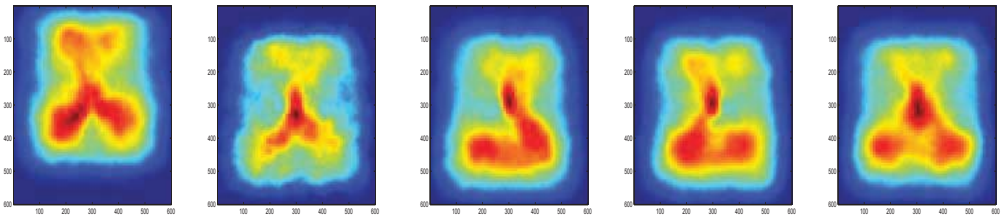
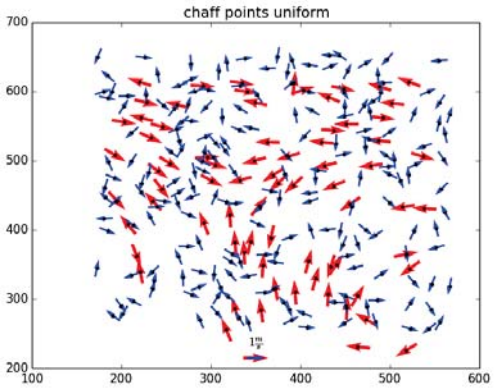


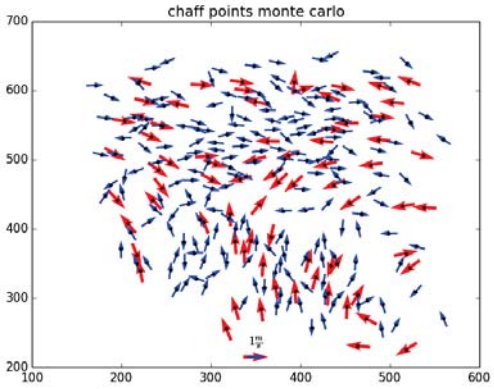
Figure 4.1: The distribution of fingerprint minutiae points in different types of fingerprints [19].

In the naive model of fingerprint randomness, it is assumed that both minutiae points and orientation is distributed uniformly. This assumption is used extensively in the literature. However, this naive approach has been shown to be not correct and has the potential to lead to sub-par detection rates [19]. Fig. 4.1 shows the heat map of distribution of fingerprint minutiae points for different types of fingerprint. As it is clear from the figure, minutiae points have a strong tendency to form a 2 to 4 clusters on the fingerprint.

In the more accurate model of [19], it is assumed that the minutiae points of each cluster do not have any statistical correlation with other cluster. However, intra-cluster correlation between the minutiae points do exist. In the model, minutiae points are model to have a multivariate Gaussian distribution and minutiae orientations are shown to abide by von Mises/Tikhonov distributions [57].



(a) Naive approach of distribution



(b) Chaff point generation using mixture model

Figure 4.2: Naive Chaff/fake minutiae point generation vs. mixture model based point generation.

4.3 Bozorth matching algorithm

Bozorth is a matching algorithm developed and implemented by NIST [65, 131], which its implementation is readily available to the public. Its attractiveness stems from its robustness to rotation and displacement of fingerprint images. This characteristics is specially very attractive for special cases that chaff points are added to the fingerprint.

Generally speaking the algorithm consists of three steps:

1. For each fingerprint, create a table of comparison based on the location of its minutiae.

2. For each potential fingerprint match, construct a compatibility table based on the relative location of their minutiae points.

3. Traverse the tables constructed in the second step, to confirm or deny a match.

The algorithm can be used for both identification and search purposes. However, the second step of the algorithm implies that its overhead will substantially increase when one searches fingerprints within a large database. For these special cases, other optimized algorithms should be used [78].

4.4 Distributed storage systems for fingerprints

We propose two methods to secure fingerprint biometric. Both of them are inspired by a recently proposed protocol for distributed password storage systems. This protocol is called Honeywords [50] and employs a two-server distributed storage mechanism to reduce the risk of server compromise. In this protocol, several fake hashed passwords for each user is stored in one server, and the other server stores the index of correct hashed password for each user. Using this backward compatible method, adversary can no longer compromise one server and do offline brute-force attacks against the password storage.

This protocol assumes that the second server has the ability to monitor the ongoing verification requests from the first server and can raise the alarm if suspicious activities are

detected. This system also requires a secure connection between the first and second server and it is implicitly assumed that the second server is less likely to be compromised, due to the fact that adversary does not have a direct connection to it. This protocol requires new randomization of indices after each authentication attempt.

In the first approach, we store one fingerprint for each instance. However, we add chaff points [51] to each fingerprint to render them useless in case of a server compromise. We would then store the correct location of minutiae points on the second server. An honest party is expected to be able to find enough number of minutiae points before being authenticated. It is expected that the added chaff points would decrease the false positive and false negative rates of the system. However, our analysis shows that the Bozorth algorithm is sufficiently robust against the added chaff points.

The security of this system depends on our ability to generate chaff points that cannot be detected from real points. Therefore, chaff points should be generated based on accurate models of fingerprint minutiae point. Fig. 4.2 shows the chaff/honey minutiae point generation based on naive and clustered Gaussian mixture model. In our example, we added 200 chaff points in a fingerprint that has 20 real ones. Based on the naive uniform chaff point generation, the probability of a successful brute force attack by an adversary is estimated to be 1.05×10^{-9}, while a brute force attack on a the Gaussian-mixture chaff generation is estimated to be 4.8×10^{-10}. This implies a one order of magnitude improvement in security of fingerprint storage units using the Gaussian-mixture model. To the best of our knowledge, the security effects of minutiae point models on the security performance of biometric storage has not been studies before this work.

In the second approach inspired by the Honeywords [50] work, instead of generating chaff points, we would just generate random fake wholesome-fingerprint for each individual fingerprint, and store the index of the correct fingerprint on a second server. Naively, one would expect that this approach would have almost the same performance as the original work on the passwords. However, a significant difference exists her. Matching fingerprints are noisy, while matching passwords are deterministic and has a much lower error

rate. If whole fingerprints are added for each fingerprint, both adversary and honest parties need to search in this space. This is a well-known fact that both false positive and false negative rate of these types of search systems would increase linearly with the size of database [78]. This fact would put a hard ceiling on the number of fake fingerprints that can be stored for each real one. However, the degree of freedom in generating chaff points are much higher. Therefore, the first approach of generating chaff points looks more promising for distributed fingerprint storage systems.

Chapter 5

Conclusions

This thesis discusses the use of indelible physical randomness in everyday phenomena such as biosignals, biometrics, and manufacturing process variations of integrated circuits. All of these sources of randomness can only be measured with tangible measurement noise. Our task in this project was designing efficient security protocols based on these easily available entropy sources. In the second Chapter, we addressed the problem of authenticating external medical controllers and programmers to Implantable Medical Devices using the indelible randomness of ECG biosignal. We presented the design and implementation of Heart-to-Heart (H2H), a lightweight "touch-to-access" scheme for Programmer-to-IMD authentication. The touch-to-access policy is enforced in H2H by a time-varying biometric, ECG heartbeat data. We performed new statistical analyses of the ECG data, including quantification of the error rates of high entropy bits. H2H draws on these analyses to achieve the first ECG-based authentication scheme that distinguishes honest from adversarial ECG signals in a rigorous statistical model and with a minimal false positive rate for a given false negative bound. We devised a novel cryptographic device pairing protocol for H2H that exploits ECG randomness to secure against active attacks, while satisfying the lightweight implementation requirements and noise margins for reliable authentication. In the second chapter, I proposed propose a novel robust and low-overhead Physical Unclonable Function (PUF) authentication, key exchange, and bit-commitment protocols that are

resilient against reverse-engineering attacks. The protocols are executed between a party with access to a physical PUF (Prover) and a trusted party who has access to the PUF compact model (Verifier). The proposed protocols do not follow the classic paradigm of exposing the full PUF responses or a transformation of them. Instead, random subsets of PUF response strings are sent to the Verifier so the exact position of the subset is obfuscated for the third-party channel observers. In the third chapter, the indelible randomness of fingerprint biometric is discussed and its statistical properties are analyzed. We propose to carefully use this source of randomness to safely store biometric information in multiple servers to guard against now common persistent attacks that might compromise servers. The work is inspired by a password storage protocol called Honeywords.

Bibliography

[1] A. L. Goldberger et al. PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals. *Circulation*, 101(23), 2000.

[2] F. Armknecht, R. Maes, A. Sadeghi, F.-X. Standaert, and C. Wachsmann. A formalization of the security features of physical functions. In *IEEE Symp. on Security and Privacy*, pages 397 –412, 2011.

[3] American Heart Association. Physical activity and blood pressure, 2012.

[4] Priyanka Bagade, Ayan Banerjee, Joseph Milazzo, and Sandeep KS Gupta. Protect your bsn: No handshakes, just namaste! In *Body Sensor Networks (BSN), 2013 IEEE International Conference on*, pages 1–6, 2013.

[5] M. Baldi, F. Chiaraluce, N. Boujnah, and R. Garello. On the autocorrelation properties of truncated maximum-length sequences and their effect on the power spectrum. *IEEE Trans. on Signal Processing*, 58, 2010.

[6] S.D. Bao, C.C.Y. Poon, Y.T. Zhang, and L.F. Shen. Using the timing information of heartbeats as an entity identifier to secure body sensor network. *IEEE Trans. on Info. Tech. in Biomedicine*, 12(6):772–779, 2008.

[7] E. Barker and A. Roginsky. Transitions: Recommendation for transitioning the use of cryptographic algorithms and key lengths. *NIST Special Publication*, 800:131A, 2011.

[8] BBC. Linkedin wins dismissal of lawsuit seeking damages for massive password breach. `http://tinyurl.com/kxp3elf`, 2012.

[9] Georg T. Becker. On the pitfalls of using arbiter-pufs as building blocks. *IACR Cryptology ePrint Archive*, 2014:532, 2014.

[10] Nathan Beckmann and Miodrag Potkonjak. Hardware-based public-key cryptography with public physically unclonable functions. In *Information Hiding*, pages 206–220. Springer, 2009.

[11] M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated key exchange secure against dictionary attacks. In *Eurocrypt*, pages 139–155, 2000.

[12] L. Biel, O. Pettersson, L. Philipson, and P. Wide. Ecg analysis: a new approach in human identification. *Instrumentation and Measurement, IEEE Transactions on*, 50:808 –812, 2001.

[13] C. Bösch, J. Guajardo, A. Sadeghi, J. Shokrollahi, and P. Tuyls. Efficient helper data key extractor on FPGAs. In *Cryptographic Hardware and Embedded Systems*, pages 181–197, 2008.

[14] R. Bousseljot, D. Kreiseler, and A. Schnabel. Nutzung der ekg-signaldatenbank cardiodat der ptb über das internet. *Biomedizinische Technik Biomedical Engineering*, 40(s1):317–318, 2009.

[15] V. Boyko, P. MacKenzie, and S. Patel. Provably secure password-authenticated key exchange using diffie-hellman. In *Advances in Cryptology*, pages 156–171. Springer, 2000.

[16] K.A. Brownley, B.E. Hurwitz, and N. Schneiderman. Cardiovascular psychophysiology. *Handbook of psychophysiology*, 2:224–264, 2000.

[17] R. Canetti and M. Fischlin. Universally composible commitments. In *Crypto*, pages 332–349, 2001.

[18] Sang-Yoon Chang, Yih-Chun Hu, Hans Anderson, Ting Fu, and Evelyn YL Huang. Body area network security: robust key establishment using human body channel. In *Proceedings of the USENIX conference on Health Security and Privacy*, pages 5–5, 2012.

[19] Yi Chen and Anil K Jain. Beyond minutiae: A fingerprint individuality model with pattern, ridge and pore features. In *Advances in Biometrics*, pages 523–533. Springer, 2009.

[20] S. Cherukuri, K.K. Venkatasubramanian, and S.K.S. Gupta. Biosec: a biometric based approach for securing communication in wireless networks of biosensors implanted in the human body. In *Parallel Processing Workshop*, pages 432–439, 2003.

[21] K. Cho and D. Lee. Biometric based secure communications without pre-deployed key for biosensor implanted in body sensor networks. In *Information Security Applications*, pages 203–218, 2012.

[22] C. Cremers, K.B. Rasmussen, B. Schmidt, and S. Capkun. Distance hijacking attacks on distance bounding protocols. In *IEEE Symp. on Security and Privacy*, pages 113–127, 2012.

[23] Foad Dabiri and Miodrag Potkonjak. Hardware aging-based software metering. In *Proc. of the Conf. on Design, Automation and Test in Europe*, pages 460–465, 2009.

[24] B. Danev, D. Zanetti, and S. Capkun. On physical-layer identification of wireless devices. *ACM Computing Surveys*, 2011.

[25] Jeroen Delvaux, Dawu Gu, Dries Schellekens, and Ingrid Verbauwhede. Secure lightweight entity authentication with strong pufs: Mission impossible? In *Cryptographic Hardware and Embedded Systems–CHES 2014*, pages 451–475. Springer, 2014.

[26] Jeroen Delvaux and Ingrid Verbauwhede. Fault injection modeling attacks on 65nm arbiter and ro sum pufs via environmental changes. Cryptology ePrint Archive: Report 2013/619, 2013. https://eprint.iacr.org/2013/619.

[27] T. Denning, K. Fu, and T. Kohno. Absence makes the heart grow fonder: New directions for implantable medical device security. In *USENIX HotSec*, 2008.

[28] distributed.net. Project RC5. http://www.distributed.net/RC5, Referenced 2012.

[29] 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.

[30] Yevgeniy Dodis, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM Journal on Computing*, 38(1):97–139, 2008.

[31] T. Drew and M. Gini. Implantable medical devices as agents and part of multiagent systems. In *Int. Joint Conf. on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1534–1541, 2006.

[32] Saar Drimer, Tim Guneysu, and Christof Paar. DSPs, BRAMs, and a pinch of logic: Extended recipes for aes on fpgas. *ACM Trans. on Reconfigurable Technology and Systems*, 3(1):3, 2010.

[33] Nourhene Ellouze, Slim Rekhis, Mohamed Allouche, and Noureddine Boudriga. Digital investigation of security attacks on cardiac implantable medical devices. *arXiv preprint arXiv:1410.4303*, 2014.

[34] K. Fu. Inside risks: Reducing risks of implantable medical devices. *Communications of the ACM*, 52(6):25–27, June 2009.

[35] B. Gassend, D. Clarke, M. Van Dijk, and S. Devadas. Delay-based circuit authentication and applications. In *Proc. of ACM Symp. on Applied Computing*, pages 294–301, 2003.

[36] Blaise Gassend. Physical Random Functions. Master's thesis, Massachusetts Institute of Technology, jan 2003.

[37] Blaise Gassend, Dwaine Clarke, Marten van Dijk, and Srinivas Devadas. Silicon physical random functions. In *Computer and Communication Security Conference*, pages 148–160, 2002.

[38] A. L. Goldberger, D. R. Rigney, and B. J. West. Chaos and fractals in human physiology. *Scientific American*, 262:42–49, 1990.

[39] S. Gollakota, H. Hassanieh, B. Ransford, D. Katabi, and K. Fu. They can hear your heartbeats: non-invasive security for implantable medical devices. In *ACM SIGCOMM*, pages 2–13, 2011.

[40] A.J. Greenspon, J.D. Patel, E. Lau, J.A. Ochoa, D.R. Frisch, R.T. Ho, B.B. Pavri, and S.M. Kurtz. 16-year trends in the infection burden for pacemakers and implantable cardioverter-defibrillators in the united states: 1993 to 2008. *Journal of the American College of Cardiology*, 58(10):1001 – 1006, 2011.

[41] Shai Halevi and Silvio Micali. Practical and provably-secure commitment schemes from collision-free hashing. In *Advances in CryptologyCRYPTO96*, pages 201–215. Springer, 1996.

[42] D. Halperin, T.S. Heydt-Benjamin, B. Ransford, S.S. Clark, B. Defend, W. Morgan, K. Fu, T. Kohno, and W.H. Maisel. Pacemakers and implantable cardiac defibrillators: Software radio attacks and zero-power defenses. In *IEEE Symp. on Security and Privacy (S& P)*, pages 129–142, 2008.

[43] D. Halperin, T. Kohno, T.S. Heydt-Benjamin, K. Fu, and W.H. Maisel. Security and privacy for implantable medical devices. *IEEE Pervasive Computing*, 7(1):30–39, Jan.-Mar. 2008.

[44] K. E. Hanna, F. J. Manning, P. Bouxsein, and A. Pope, editors. *Innovation and Invention in Medical Devices: Workshop Summary*. The National Academies Press, 2001.

[45] Lin Hong, Yifei Wan, and Anil Jain. Fingerprint image enhancement: algorithm and performance evaluation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20(8):777–789, 1998.

[46] Chunqiang Hu, Xiuzhen Cheng, Fan Zhangand, Dengyuan Wuand, Xiaofeng Liao, and Dechang Chen. OPFKA: Secure and efficient ordered-physiological-feature-based key agreement for wireless body area networks. In *Proc. of IEEE INFOCOM, To Appear*, 2013.

[47] F. Hu, Q. Hao, M. Lukowiak, Q. Sun, K. Wilhelm, S. Radziszowski, and Y. Wu. Trustworthy data collection from implantable medical devices via high-speed security implementation based on IEEE 1363. *IEEE Trans. on Info. Tech. in Biomedicine*, 14(6):1397–1404, Nov. 2010.

[48] Anil K Jain and Karthik Nandakumar. Biometric authentication: System security and user privacy. *IEEE Computer*, 45(11):87–92, 2012.

[49] A. Juels and M. Wattenberg. A fuzzy commitment scheme. In *ACM CCS*, pages 28–36, 1999.

[50] Ari Juels and Ronald L Rivest. Honeywords: Making password-cracking detectable. In *Proceedings of the ACM SIGSAC conference on Computer & communications security*, pages 145–160. ACM, 2013.

[51] Ari Juels and Madhu Sudan. A fuzzy vault scheme. *Designs, Codes and Cryptography*, 38(2):237–257, 2006.

[52] Deniz Karakoyunlu and Berk Sunar. Differential template attacks on puf enabled cryptographic devices. *Intl. Workshop on Information Forensics and Security*, pages 1–6, 2010.

[53] D. Karaouglan and A. Levi. A survey on the development of security mechanisms for body area networks. *The Computer Journal*, 2013.

[54] Stefan Katzenbeisser, Ünal Kocabaş, Vladimir Rožić, Ahmad-Reza Sadeghi, Ingrid Verbauwhede, and Christian Wachsmann. PUFs: Myth, fact or busted? a security evaluation of physically unclonable functions (PUFs) cast in silicon. In *Cryptographic Hardware and Embedded Systems*, pages 283–301, 2012.

[55] C. Kaufman, R. Perlman, and M. Speciner. *Network security: private communication in a public world*. Prentice Hall Press, 2002.

[56] S. S. Keller. NIST-recommended random number generator based on ANSI X9.31 appendix A.2.4 using the 3-key triple DES and AES algorithms. Technical report, NIST, 2005.

[57] CG Khatri and KV Mardia. The von mises-fisher matrix distribution in orientation statistics. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 95–106, 1977.

[58] Mooseop Kim, Jaecheol Ryou, and Sungik Jun. Efficient hardware architecture of SHA-256 algorithm for trusted mobile computing. In *Information Security and Cryptology*, pages 240–252, 2009.

[59] Cetin Kaya Koc, editor. *Cryptographic Engineering*. Springer, 1 edition, 2008.

[60] Farinaz Koushanfar. *Hardware Metering: A Survey*, pages 103–122. Springer, 2011.

[61] Chunxiao Li, Anand Raghunathan, and Niraj K Jha. Hijacking an insulin pump: Security attacks and defenses for a diabetes therapy system. In *IEEE Int. Conf. on e-Health Networking Applications and Services*, pages 150–156, 2011.

[62] Xiaofeng Liao. Body area network security: A fuzzy attribute-based signcryption scheme. *IEEE Journal on Selected Areas in Communications, To Appear*, 2013.

[63] Daihyun Lim. Extracting secret keys from integrated circuits. Master's thesis, Massachusetts Institute of Technology, 2004.

[64] MISRA Limited. Misra-c:2004 - guidelines for the use of the c language in critical systems. Technical report, Motor Industry Software Reliability Association, 2004.

[65] Sainath Maddala, Josef Ström Bartunek, and Mikael Nilsson. Implementation and evaluation of nist biometric image software for fingerprint recognition. In *Signal and Image Processing (ICSIP), 2010 International Conference on*, pages 207–211. IEEE, 2010.

[66] 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*, pages 332–347, 2009.

[67] R. Maes and I. Verbauwhede. Physically unclonable functions: a study on the state of the art and future research directions. In A.-R. Sadeghi and D. Naccache, editors, *Towards Hardware-Intrinsic Security*. Springer, 2010.

[68] Ahmed Mahmoud, Ulrich Rhrmair, Mehrdad Majzoobi, and Farinaz Koushanfar. Combined Modeling and Side Channel Attacks on Strong PUFs. Cryptology ePrint Archive, Report 2013/632, 2013. `http://eprint.iacr.org/`.

[69] Ahmed Mahmoud, Ulrich Ruhrmair, Mehrdad Majzoobi, and Farinaz Koushanfar. Combined modeling and side channel attacks on strong pufs. Cryptology ePrint Archive: Report 2013/632, 2013. `https://eprint.iacr.org/2013/632`.

[70] W.H. Maisel. Safety issues involving medical devices. *Journal of the American Medical Association*, 294(8):955–958, Aug. 2005.

[71] M. Majzoobi, G. Ghiaasi, F. Koushanfar, and S.R. Nassif. Ultra-low power current-based PUF. In *IEEE Int. Symp. on Circuits and Systems*, pages 2071–2074. IEEE, 2011.

[72] M. Majzoobi and F. Koushanfar. Time-bounded authentication of FPGAs. In *Under Revision for IEEE Trans. on Information Forensics and Security (TIFS)*, 2011.

[73] M. Majzoobi, F. Koushanfar, and S. Devadas. FPGA-based true random number generation using circuit metastability with adaptive feedback control. *Cryptographic Hardware and Embedded Systems*, pages 17–32, 2011.

[74] M. Majzoobi, F. Koushanfar, and M. Potkonjak. Lightweight secure PUF. In *Int. Conf. on Computer Aided Design*, pages 670–673, 2008.

[75] M. Majzoobi, F. Koushanfar, and M. Potkonjak. Testing techniques for hardware security. In *International Test Conference*, pages 1–10, 2008.

[76] 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.

[77] Mehrdad Majzoobi, Farinaz Koushanfar, and Srinivas Devadas. FPGA PUF using programmable delay lines. In *IEEE Int. Workshop on Information Forensics and Security*, pages 1–6. IEEE, 2010.

[78] Davide Maltoni, Dario Maio, Anil K Jain, and Salil Prabhakar. *Handbook of fingerprint recognition*. springer, 2009.

[79] R.P. McEvoy, F.M. Crowe, C.C. Murphy, and W.P. Marnane. Optimisation of the SHA-2 family of hash functions on FPGAs. In *Emerging VLSI Technologies and*

Architectures, 2006. IEEE Computer Society Annual Symposium on, pages 6–pp. IEEE, 2006.

[80] Saro Meguerdichian and Miodrag Potkonjak. Matched public PUF: ultra low energy security platform. In *Proc. of Int. Symp. on Low Power Electronics and Design*, pages 45–50, 2011.

[81] Saro Meguerdichian and Miodrag Potkonjak. Using standardized quantization for multi-party ppuf matching: Foundations and applications. In *Int. Conference on Computer-Aided Design*, pages 577–584, 2012.

[82] M. Meingast, T. Roosta, and S. Sastry. Security and privacy issues with health care information technology. In *IEEE Engineering in Medicine and Biology Society*, pages 5453–5458, 2006.

[83] Dominik Merli, Dieter Schuster, Frederic Stumpf, and Georg Sigl. Side-channel analysis of pufs and fuzzy extractors. In *Trust and Trustworthy Computing*, pages 33–47. Springer, 2011.

[84] G.B. Moody and R.G. Mark. The impact of the MIT-BIH arrhythmia database. *IEEE Engineering in Medicine and Biology Magazine*, 20(3):45–50, 2001.

[85] J. H. Nagel, K. Han, B. E. Hurwitz, and N. Schneiderman. Assessment and diagnostic applications of heart rate variability. *Biomedical engineering-applications, basis & communications*, 5:147–158, 1993.

[86] J. Neyman and E.S. Pearson. On the problem of the most efficient tests of statistical hypotheses. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 231:289–337, 1933.

[87] E. Oztürk, G. Hammouri, and B. Sunar. Towards robust low cost authentication for pervasive devices. In *Pervasive Computing and Communications*, pages 170–178, 2008.

[88] C. Paar, J. Pelzl, and B. Preneel. *Understanding cryptography: a textbook for students and practitioners*. Springer, 2010.

[89] Z. Paral and S. Devadas. Reliable and efficient PUF-based key generation using pattern matching. In *Int. Symp. on Hardware-Oriented Security and Trust*, pages 128 –133, 2011.

[90] pcworld.com. Linkedin wins dismissal of lawsuit seeking damages for massive password breach. `http://tinyurl.com/k7r6hqr`, 2013.

[91] E. Pino, L. Ohno-Machado, E. Wiechmann, and D. Curtis. Real–time ecg algorithms for ambulatory patient monitoring. In *AMIA Annual Symposium Proceedings*, volume 2005, page 604, 2005.

[92] M.Z. Poh, D.J. McDuff, and R.W. Picard. Non-contact, automated cardiac pulse measurements using video imaging and blind source separation. *Optics Express*, 18:10762–10774, 2010.

[93] C.C.Y. Poon, Y.T. Zhang, and S.D. Bao. A novel biometrics method to secure wireless body area sensor networks for telemedicine and m-health. *IEEE Communications Magazine*, 44(4):73–81, 2006.

[94] Miodrag Potkonjak, Saro Meguerdichian, Ani Nahapetian, and Sheng Wei. Differential public physically unclonable functions: architecture and applications. In *Design Automation Conference (DAC)*, pages 242–247, 2011.

[95] M. Poturalski, M. Flury, P. Papadimitratos, J.-P. Hubaux, and J.-Y. Le Boudec. Distance bounding with ieee 802.15.4a: Attacks and countermeasures. *IEEE Trans. on Wireless Comms.*, 10(4):1334– 1344, 2011.

[96] Jeyavijayan Rajendran, Garrett S Rose, Ramesh Karri, and Miodrag Potkonjak. Nano-PPUF: A memristor-based security primitive. In *Computer Society Annual Symp. on VLSI*, pages 84–87, 2012.

[97] K.B. Rasmussen, C. Castelluccia, T.S. Heydt-Benjamin, and S. Capkun. Proximity-based access control for implantable medical devices. In *Proc. of Computer and communications security*, pages 410–419, 2009.

[98] P. S. Ravikanth, B. Recht, J. Taylor, and N. Gershenfeld. Physical one-way functions. *Science*, 297:2026–2030, 2002.

[99] Garrett S Rose, Jeyavijayan Rajendran, Nathan R McDonald, Ramesh Karri, Miodrag Potkonjak, and Bryant T Wysocki. Hardware security strategies exploiting nanoelectronic circuits. In *ASP-DAC*, pages 368–372, 2013.

[100] M. Rostami, W. Burleson, F. Koushanfar, and A. Juels. Balancing security and utility in medical devices? In *Proc. of Design Automation Conference*, pages 1–6, 2013.

[101] Masoud Rostami, Farinaz Koushanfar, and Ramesh Karri. A primer on hardware security: Models, methods, and metrics. *Proceedings of the IEEE*, 102(8):1283–1295, 2014.

[102] Masoud Rostami, James B Wendt, Miodrag Potkonjak, and Farinaz Koushanfar. Quo vadis, PUF? In *Design, Automation & Test in Europe*, 2014.

[103] Masoud Rostami, James B Wendt, Miodrag Potkonjak, and Farinaz Koushanfar. Quo vadis, puf?: trends and challenges of emerging physical-disorder based security. In *Proceedings of the conference on Design, Automation & Test in Europe*, page 352. European Design and Automation Association, 2014.

[104] RSA, The Security Division of EMC. RSA SecurID authentication in action: Securing privileged user access, 2007.

[105] U. Rührmair, F. Sehnke, J. Sölter, G. Dror, S. Devadas, and J. Schmidhuber. Modeling attacks on physical unclonable functions. In *ACM Conf. on Computer and Communications Security*, pages 237–249, 2010.

[106] Ulrich Rührmair. Simpl systems: On a public key variant of physical unclonable functions. *IACR Cryptology ePrint Archive*, 2009:255, 2009.

[107] Ulrich Rührmair. Oblivious transfer based on physical unclonable functions. In *Trust and Trustworthy Computing*, pages 430–440. Springer, 2010.

[108] Ulrich Rührmair. SIMPL systems, or: can we design cryptographic hardware without secret key information? In *SOFSEM 2011: Theory and Practice of Computer Science*, pages 26–45. Springer, 2011.

[109] Ulrich Rührmair, Srinivas Devadas, and Farinaz Koushanfar. Security based on physical unclonability and disorder. *Book Chapter in Introduction to Hardware Security and Trust*, 2011.

[110] Ulrich Ruhrmair, Srinivas Devadas, and Farinaz Koushanfar. *Security based on Physical Unclonability and Disorder*, pages 65–102. Springer, 2011.

[111] Ulrich Ruhrmair, Jan Solter, Frank Sehnke, Xiaolin Xu, Ahmed Mahmoud, Vera Stoyanova, Gideon Dror, Jurgen Schmidhuber, Wayne Burleson, and Srinivas Devadas. PUF modeling attacks on simulated and silicon data. *IEEE Trans. on Information Forensics and Security*, pages 1876 – 1891, 2013.

[112] Ulrich Rührmair and Marten van Dijk. On the practical use of physical unclonable functions in oblivious transfer and bit commitment protocols. *Journal of Cryptographic Engineering*, pages 1–12, 2013.

[113] 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.

[114] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, and E. Barker. A statistical test suite for random and pseudorandom number generators for cryptographic applications. Technical report, DTIC Document, 2001.

[115] Michael Rushanan, Aviel D Rubin, Denis Foo Kune, and Colleen M Swanson. Sok: Security and privacy in implantable medical devices and body area networks. In *Proceedings of ACM SIGSAC conference on Computer & communications security*, 2014.

[116] Michael Rushanan, Aviel D Rubin, Denis Foo Kune, and Colleen M Swanson. Sok: Security and privacy in implantable medical devices and body area networks. In *Proc. of Computer and communications security*, 2014.

[117] Claude E Shannon. Communication theory of secrecy systems. *Bell system technical journal*, 28(4):656–715, 1949.

[118] Wenbo Shen, Peng Ning, Xiaofan He, and Huaiyu Dai. Ally friendly jamming: How to jam your enemy and maintain your own wireless connectivity at the same time. In *IEEE Symp. on Security and Privacy*, 2013.

[119] M.G. Signorini, F. Marchetti, and S. Cerutti. Applying nonlinear noise reduction in the analysis of heart rate variability. *Engineering in Medicine and Biology Magazine, IEEE*, 20(2):59–68, 2001.

[120] F. Stajano and R.J. Anderson. The resurrecting duckling: Security issues for ad-hoc wireless networks. In *Int. Workshop of Security Protocols*, pages 172–194, 1999.

[121] G. Suh and S. Devadas. Physical unclonable functions for device authentication and secret key generation. In *Design Automation Conference*, pages 9–14, 2007.

[122] B. Sunar, W.J. Martin, and D.R. Stinson. A provably secure true random number generator with built-in tolerance to active attacks. *IEEE Trans. on Computers*, 56(1):109–119, 2007.

[123] LA Times. Internet users advised to change passwords due to 'heartbleed' bug. `http://tinyurl.com/lod868z`, 2014.

[124] K.K. Venkatasubramanian and S.K.S. Gupta. Physiological value-based efficient usable security solutions for body sensor networks. *ACM Trans. Sensor Networks*, 6(4):31:1–31:36, July 2010.

[125] Krishna K Venkatasubramanian, Ayan Banerjee, and S Gupta. Plethysmogram-based secure inter-sensor communication in body area networks. In *IEEE Military Communications Conference*, pages 1–7, 2008.

[126] Krishna K Venkatasubramanian, Ayan Banerjee, and Sandeep Kumar S Gupta. PSKA: usable and secure key agreement scheme for body area networks. *IEEE Trans. on Information Technology in Biomedicine*, 14(1):60–68, 2010.

[127] S. Warren, J. Lebak, J. Yao, J. Creekmore, A. Milenkovic, and E. Jovanov. Interoperability and security in wireless body area network infrastructures. In *IEEE Engineering in Medicine and Biology Society*, pages 3837–3840, 2005.

[128] JP Welch, PJ Ford, RS Teplick, and RM Rubsamen. The massachusetts general hospital-marquette foundation hemodynamic and electrocardiographic database – comprehensive collection of critical care waveforms. *Clinical Monitoring*, 7(1):96–97, 1991.

[129] James B Wendt and Miodrag Potkonjak. Nanotechnology-based trusted remote sensing. In *IEEE Sensors*, pages 1213–1216, 2011.

[130] James B Wendt and Miodrag Potkonjak. The bidirectional polyomino partitioned PPUF as a hardware security primitive. In *IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, 2013.

[131] Charles L Wilson, Craig I Watson, Michael D Garris, and Austin Hicklin. *Studies of fingerprint matching using the NIST verification test bed (VTB)*. US Department of Commerce, Technology Administration, National Institute of Standards and Technology, 2003.

[132] F. Xu, Z. Qin, C.C. Tan, B. Wang, and Q. Li. IMDGuard: Securing implantable medical devices with the external wearable guardian. In *Proc. of IEEE INFOCOM*, pages 1862–1870, 2011.

[133] Teng Xu, James B Wendt, and Miodrag Potkonjak. Digital bimodal function: an ultra-low energy security primitive. In *Proc. of Int. Symp. on Low Power Electronics and Design*, pages 292–296, 2013.

[134] 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.

[135] R. Yulmetyev, P. Hänggi, and F. Gafarov. Quantification of heart rate variability by discrete nonstationary non-Markov stochastic processes. *Physical Review E*, 65(4):046107, 2002.

[136] K. Zetter. DigiNotar files for bankruptcy in wake of devastating hack. *Wired*, 20 Sept. 2011.

[137] Thoams Guthrie Zimmerman. Personal area networks: near-field intrabody communication. *IBM systems Journal*, 35:609–617, 1996.

[138] Thomas G Zimmerman, Joshua R Smith, Joseph A Paradiso, David Allport, and Neil Gershenfeld. Applying electric field sensing to human-computer interfaces. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 280–287, 1995.