January 2007

Randomness in Integrated Circuits with Applications in Device Identification and Random Number Generation

Daniel Holcomb
University of Massachusetts Amherst, dholcomb@ecs.umass.edu

RANDOMNESS IN INTEGRATED CIRCUITS WITH APPLICATIONS IN DEVICE IDENTIFICATION AND RANDOM NUMBER GENERATION

A Thesis Presented

by

DANIEL E. HOLCOMB

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL AND COMPUTER ENGINEERING

September 2007

Electrical and Computer Engineering

RANDOMNESS IN INTEGRATED CIRCUITS WITH APPLICATIONS IN DEVICE IDENTIFICATION AND RANDOM NUMBER GENERATION

A Thesis Presented

by

DANIEL E. HOLCOMB

Approved as to style and content by:

_____
Wayne P. Burleson, Chair

_____
Kevin Fu, Member

_____
Israel Koren, Member

_____
C.V. Hollot, Department Head
Department of Electrical and Computer Engineering

ACKNOWLEDGEMENTS

I acknowledge and thank the following people for their contributions to this work. I thank my advisor Professor Wayne Burleson for teaching me integrated circuits, motivating me and guiding my research. Professor Kevin Fu for introducing me to both security and RFID, and for his considerable help and enthusiasm in driving this work. Professor Israel Koren for serving on my thesis committee. Professor Adam Stubblefield of Computer Science at Johns Hopkins University and Dr. Ari Juels of RSA Labs for providing feedback on early versions of the work, and Adam for his pointers on how to generate random numbers. Intel Research and Joshua R. Smith for providing and supporting the WISP platforms. Professor Sandip Kundu for his help with understanding the costs of manufacturing integrated circuits. I thank all of my collaborators in RFID-CUSP and VLSI Circuits and Systems Group for their input, feedback, and help that made the work possible. Outside of the scope of this thesis, I thank my family, and the faculty, staff, and students of UMass Amherst, who have given me great support throughout my many years in Amherst. This material is based upon work supported by the National Science Foundation under Grant No. 0627529.

ABSTRACT

RANDOMNESS IN INTEGRATED CIRCUITS WITH APPLICATIONS IN DEVICE IDENTIFICATION AND RANDOM NUMBER GENERATION

SEPTEMBER 2007
DANIEL E. HOLCOMB
B.S., UNIVERSITY OF MASSACHUSETTS AMHERST
M.S.E.C.E., UNIVERSITY OF MASSACHUSETTS AMHERST
Directed by: Professor Wayne P. Burleson

RFID applications create a need for low-cost security and privacy in potentially hostile environments. To accomplish these goals of security and privacy, static identifiers and random numbers are required. Motivated by cost constraints, this thesis explores generating both identifiers and random numbers from existing CMOS circuitry, using a system of Fingerprint Extraction and Random Numbers in SRAM (FERNS). The identity results from the impact of manufacture-time physically random device threshold mismatch on the initial state of SRAM, and the randomness results from the impact of run-time physically random noise. FERNS is supported by an analytical model of the relative impacts of process variation and noise, and by experimental data from virtual tags, microcontroller memory, and the WISP UHF passive RFID tag. It is shown that virtual tags can be uniquely identified amongst a population of 160 using less than 50 bits of SRAM with an efficient matching algorithm. It is shown that a 128 bit true random number capable of passing statistical tests can be extracted from 256 bytes of SRAM. Based on these results and the observation that FERNS is well suited to passive applications, we conclude that FERNS is a viable approach to both identification and true random number generation in RFID tags.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

CHAPTER 1

INTRODUCTION

1.1 Randomness in Integrated Circuits

Randomness due to process variation and noise exists in all integrated circuits. This randomness causes discrepancies across circuits and across time, and can impact computation. In large scale digital circuits, design steps are taken to isolate the computation from this randomness by using noise margins and timing margins. Small scale integrated circuits such as radio frequency identification (RFID) tags and smartcards are quite different from VLSI circuits. These circuits are extremely cost-constrained and operate in potentially hostile environments. In order to attain an acceptable level of security and privacy, low-cost implementations of true random number generation (TRNG) and chip identification (ID) circuits are needed.

While TRNG and ID are typically accomplished independently, this thesis explores the possibility of accomplishing both TRNG and ID using shared circuitry. Further, it is shown that this shared circuitry need not be dedicated, but can be common CMOS Static Random Access Memory (SRAM). The reuse of SRAM allows overhead to be minimized. This thesis makes several contributions to the field of secure hardware:

1 – Demonstrating that SRAM initialization generates usable identifying characteristics

2 – Demonstrating that SRAM initialization generates true randomness

1.2 RFID Technology

RFID tags come in a variety of specifications. Low frequency (LF) tags operate at 125-134 kHz and 140-148.5 kHz, high-frequency (HF) tags operate at 13.56 MHz, and ultra-high-frequency (UHF) tags operate at 902-928 MHz in the United States.

LF tags are covered by the ISO 11784, 11785 and 18000-2 standards. LF tags require larger antennas due to their long wavelength, and are typically not found in disposable transponders, but instead in reusable applications such as automobile immobilizers and the Exxon Mobil SpeedPass™ system. LF

tags can penetrate most materials, including water and body tissue, allowing applications in animal tracking.

HF tags are covered by ISO 14443, 15693, and 18000-3 standards. Using smaller antennas than LF tags, typically uses for HF tags include RFID credit cards [1], NXP Semiconductors' MIFARE cards [2], and E-Passports [3]. Compared to LF tags, HF tags are lower cost, and offer better throughput. With higher throughput, multiple tags can be read at once, giving rise to a need for anti-collision protocols. Read ranges for LF tags are typically limited to around 1m. Typical cost for an HF tag is around $1.

UHF tags are a more recent development, described by EPC global class 1 gen 2 protocol which was later ratified as ISO 18000-6C [4]. UHF tags communicate using backscatter instead of the inductive coupling employed in LF and HF tags. Backscatter provides a greater read range, on the order of 3-6m. UHF tags do not work well in the presence or liquids of metals. A typical UHF tag has only a 96 bit serial number and no memory. This minimal functionality allows manufacture at low cost, currently below 15 cents and predicted to go below 10 cents per tag by 2008 [5]. Because of the low-cost and significant read-range, UHF tags have great potential in supply chain management, and are already in use by Wal-Mart and Gillette among others.

Across all specifications, passive RFID applications present circuit designers with a unique set of challenges. With no on-board power source on tags, power consumption must be minimized. Because little performance is needed, and to meet pricing pressures, area and complexity must also be minimized.

1.2.1 Power Constraints

In passive tags, all power used by the tag is harvested via coupling to the reader allowing only minute power consumption. The coupling can either be near field or far field. Near field systems use inductive coupling between the tag and reader, where the tag harvests reactive energy that is circulating around the reader antenna. The name "near field" indicates that the energy drops off according to the cube of the distance between the tag and reader. Far field systems couple to real power in free-space-propagating electromagnetic plane waves, and drops off according to the square of the distance between tag and reader. This accounts for the difference in read ranges between the different specifications; LF and HF tags typically use near field, while the longer range of UHF is enabled by its use of far field [6].

Regardless of the coupling mechanism employed by the tag, the lack of an onboard energy source creates a limited power budget. This is a different scenario from battery powered devices, which are constrained by energy storage. A typical tag consumes only microamperes of current at under a volt, or a few microwatts of power. One such example is an 8.9μW baseband processor using 0.35μm Chartered technology [7]. A second example is the 8μW Zuma RFID^{TM} chip in 0.25μm subthreshold CMOS technology, with complexity equivalent to an Intel 8086 [8]. Another example is a UHF core designed to run on 0.3 μW using a high V_{th}, subthreshold 0.18μ standard cell CMOS, and using multiple clock domains to allow slow energy-optimized computation where possible [9]. For an overview of several subthreshold logic styles, see [10].

1.2.2 Area and Cost Constraints

RFID circuits also face extreme cost constraints. Desirable pricing points are 5 cents for simple EPC type tags [11], and approximately 50 cents for a security enabled HF tag. With such low costs desired, both amortized non-recurring engineering (NRE) cost and incremental silicon costs must be minimized.

NRE refers to the fixed cost of creating a circuit that is irrespective of the quantity produced, including the cost of research, design, and fixed costs associated with manufacturing. Amortized NRE refers to the share of NRE that can be assigned to each chip produced, and directly influences the per-unit price of the product. Amortized NRE is minimized in two ways. First, and most obvious, is to maximize the number of chips sold, in order to minimize the share that is assigned to each chip. The second strategy is to minimize the NRE itself by avoiding cutting edge technology, and instead using technology that lags the leading edge by several generations. The advantages of this strategy are less expensive and mature process flows, lower mask costs (table 1.1), and stable commodity processes with multiple suppliers [8].

Table 1.1 Mask costs increase with technology scaling. Source ITRS design 2006 [12]

Year of Production	2005	2006	2007	2008	2009	2010	2011	2012
DRAM $^{1}/_{2}$ Pitch (nm)	80	70	65	57	50	45	40	36
% Mask cost ($m) from publicly available data	1.5	2.2	3.0	4.5	6.0	9.0	12.0	18.0

To minimize overall costs, incremental costs must also be kept to a minimum. Incremental costs are the per unit costs that are paid for producing a chip, including the costs of raw materials and operating fabrication machinery. In general terms, incremental costs are directly correlated to the silicon area required to implement the tag, and are less correlated to the technology node used. The combination of amortized NRE and incremental costs create a "sweet spot" for RFID (figure 1.1).

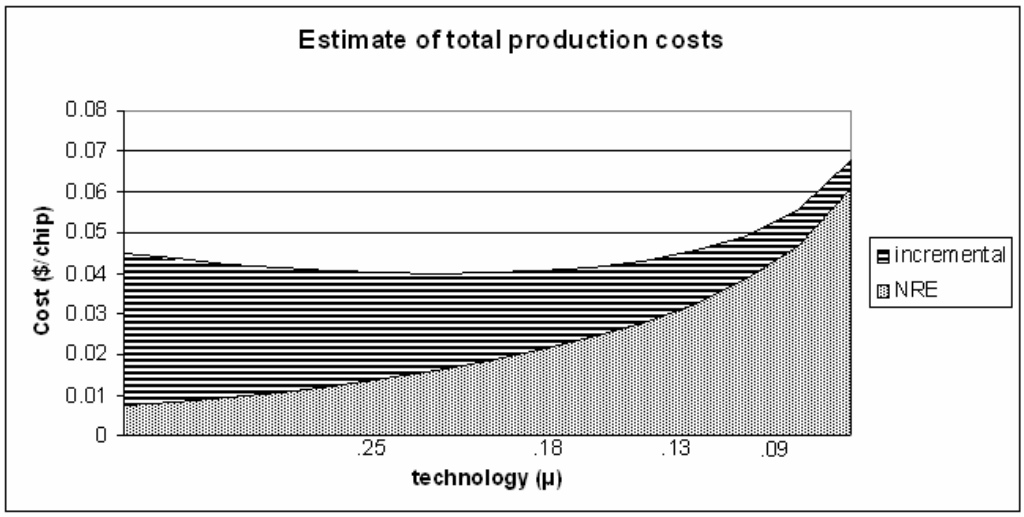

Figure 1.1: The RFID "sweet spot" is a compromise between minimizing amortized NRE and minimizing incremental costs.

Given that leading edge technologies are not economically feasible for RFID, silicon area must be minimized via low complexity. A rough estimate is that a passive RFID tag should be no bigger than $0.5mm^2$ [1] which is roughly the size of 22,000 minimum sized inverters or 3,800 asynchronous reset D flip flops in 0.18µ CMOS [8]. However, only a small fraction of the silicon area is available for digital logic. One estimate of available logic functionality is that a typical EPC class 1 tag has 1,000-4,000 gates, with class 2 tags having several thousand more [14]. The Zuma RFID™ chip fits 41,798 transistors (10,450 gate equivalents) within $.58mm^2$ [8], but inspection of the breakdown of functionality shows that only around 3,000 transistors are available for implementing logic; the remainder of silicon area is used for power rectification, storage capacitors, signal modulation and non-volatile memory, as is typical in passive RFID tags. Among the relatively few transistors available for digital logic, only 200-2,000 are typically available for security [15]. Traditional cryptographic ciphers do not fit within these constraints, leading to a wealth of research in developing secure low-cost ciphers (table 1.2).

Table 1.2: Gate counts of small footprint ciphers.

	key size	gate equivalents
Present-80 [16]	80	1570
AES-128 [17]	128	3400
HIGHT [18]	128	3048
mCrypton [19]	96	2681
DES [20]	56	2309
DESXL [20]	184	2168
Trivium [21]	80	2599
Grain [21]	80	1294

1.3 Approaches to Random Number Generation

Random numbers play an important role in cryptography; for example, they find use as cryptographic keys and one-time pads. Any predictability in random numbers will compromise the security of the scheme in which they are used. The approaches to creating random numbers can be broadly classified into two main categories, True Random Number Generation (TRNG) and Pseudo Random Number Generation (PRNG). TRNGs rely on a physically random process as a source of entropy, whereas PRNGs produce outputs that have statistical properties of random numbers, yet are fully deterministic. Because PRNGs use simple logic, they sometimes find use in RFID tags. TRNGs are desirable for security applications. The random process and how it is harvested both vary across TRNG designs. One physically random process in integrated circuits is thermal noise, which describes voltage variations that exist when a conductor is in equilibrium [22,23]. A related physically random process is shot noise, which describes the randomness in a current as it begins to flow through a conductor [24]. To create a random number from such a physically random process, a harvesting method is required. A well-known method for harvesting noise is through its manifestation in the jitter of free-running oscillators [25] and in subthreshold chaotic oscillators [26]. A second way to extract thermal or shot noise is by amplifying the noise to a measurable level, by use of direct amplification or through the large gain that is inherent in metastable CMOS devices [27,28].

1.4 Approaches to Circuit Identification

In the most general terms, RFID circuits can be identified either through the use of non-volatile memories or the use of some identifying physical characteristic, which is referred to here as fingerprinting. The non-volatile approach involves programming an identity into a tag at the time of manufacture using EPROM, EEPROM, flash, fuse, or more exotic strategies such as the electron beam programming demonstrated by the Hitachi Mu Chip in 90nm SOI [29]. While non-volatile identities are static and fully reliable, they have drawbacks. The use of non-volatile memories adds about 30 process steps in manufacturing [8]; Even if only a small amount of non-volatile storage is used, the process cost must be paid across the entire chip area. Additionally, supporting circuitry such as charge pumps for tunneling oxide devices, and programming transistors for fuse devices, are needed. Finally, non-volatile memories can reveal state to invasive attacks.

The fingerprint approach to identification uses the process variation that is inherent in the manufacture of integrated circuits for differentiation between chips. Process variation comes in many forms, including lithographic variations in effective feature size and random threshold voltages. In terms of producing identifying characteristics, it is generally not the absolute variation that matters, but instead the variation mismatch between the nearby devices that are implementing the identifying function. Lithographic variations are correlated among local devices and devices occupying the same within-field position on different chips [30]. Variations in threshold voltages are due to random fluctuations in the concentration of dopant atoms and are not spatially correlated, making an ideal identifying characteristic. With the scaling trends of integrated circuits, device threshold variability is expected to increase, implying that threshold based identification will continue to be a viable approach in the coming years (table 1.3).

Table 1.3: Variability in device threshold voltages is predicted to increase, and is due primarily to dopant concentrations. Source ITRS 2006-design[12].

Yearh of Production	2005	2006	2007	2008	2009	2010	2011	2012
DRAM $^{1}/_{2}$ Pitch (nm)	80	70	65	57	50	45	40	36
% Vth variability Doping Variability impact on Vth	24%	29%	31%	35%	40%	40%	40%	58%
% Vth variability Includes all sources	26%	29%	33%	37%	42%	42%	42%	58%

Simple physical fingerprints can be used as identifying signatures. One such circuit was demonstrated using MOS device random threshold assignment as the identifying characteristic and supporting circuitry to indirectly measure these threshold voltages [31]. A similar approach using metastability for identifying RFID tags is demonstrated by Su, Holleman and Otis [32]. The physical uncloneable function (PUF) uses a physical race condition where the racing paths are selected by the applied input [33]. The identifying output is determined by the relative delays of the two paths, enabling the PUF to work like a hash function. The advantages to using physical fingerprints are their use of ordinary CMOS process and the fact that no programming step is required. The most significant drawback to physical fingerprint identification is that the identities are impacted by noise. The PUF circuit uses noise to accomplish random number generation along with authentication [34]. The randomness is obtained by finding and then persistently applying inputs that cause races between well-matched paths, leading to each binary outcome with equal probability.

1.5 The Observed Relation between Physical ID and TRNG

Physical ID circuits use small manufacturing variations to differentiate chips. Because the manufacturing variations are small, the circuits must magnify them up to useable levels. TRNG circuits harvest a physically random process for creating random bits; because the physically random processes are small, the circuits must magnify these small variations up to a level that can be captured. This leads to the observation that both physical ID circuits and TRNG circuits are designed to be sensitive to slight variations in conditions. All physical ID circuits show some effects of noise. As was previously mentioned, PUFs have been shown to constructively use this randomness for TRNG, and the chip ID circuit of Su, Holleman and Otis also shows the effects of randomness in having 3% unstable bits [32].

Based on this observed relationship between physical ID and TRNG, the remainder of this thesis explores the fact that the initial state of static random access memory (SRAM) has the property of being a slightly random, massively parallel, physical ID. A system called fingerprint extraction and random numbers in SRAM (FERNS) outlines the usage of SRAM state as a source of both identity and randomness, targeting application in RFID circuits. Chapter 2 presents FERNS and gives its physical principles. Chapter 3 outlines how FERNS can provide identification. Chapter 4 outlines how FERNS can

provide randomness. Chapter 5 compares FERNS to existing methods of random number generation and identification, and concludes the thesis.

CHAPTER 2

FINGERPRINT EXTRACTION AND RANDOM NUMBERS IN SRAM (FERNS)

2.1 Introduction

FERNS is built upon the idea that the stabilization of each SRAM cell at power-up reveals a physical fingerprint. Uninitialized SRAM is normally considered to be in a logically unknown state. By descending below the logical level of abstraction, and considering SRAM state to be a physical fingerprint, a wealth of information is found. We propose FERNS as a means of using initial SRAM state as a source of both identifying fingerprints and true random number generation. To avoid revealing randomness and destroying its utility, bits that are used for randomness cannot be plainly transmitted as part of an ID. With an SRAM cell being the required circuitry for merely storing a bit, each SRAM cell is perhaps the smallest possible physical fingerprint capable of producing a digital output. In the remainder of this chapter we build up the FERNS system by first identifying why SRAM initialization yields a useful physical fingerprint, and then giving our methodologies for experimentally verifying the existence of physical fingerprints. The discussion of applications for the fingerprints is discussed in chapters 3 and 4.

2.2 Physical Principles of Fingerprints

The fingerprint produced by an instance of an SRAM consists of the initial values produced by all of the bits of that SRAM. The initial state of each bit is a function of both process variation and of random noise. The initial state of some bits is almost entirely dependent on process variation, while the initial states of other bits are highly dependent on noise. Many bits of SRAM depend on both variation and noise to some degree. The time-invariance of process variation gives consistency to the fingerprint, allowing it to be used for identification. Because noise conditions are different each time an SRAM is powered up, the fingerprint also captures randomness, allowing it to be used as an entropy source in TRNG.

Each bit of SRAM is a 6 transistor storage cell, consisting of cross coupled inverters and access transistors. Each of the inverters drives one of the two state nodes, labeled A and B. In the digital realm, this circuit has 4 possible states; states 01 and 10 are stable, while states 00 and 11 are unstable (figure 2.1).

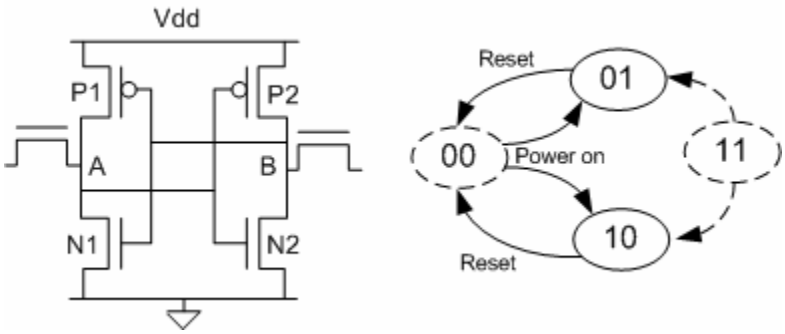

Figure 2.1: Schematic of a basic 6T SRAM cell and state diagram showing all possible states and transitions.

Consider the behavior of a bit of SRAM when power is applied to the chip. Initially, the circuit is in an unpowered state with both nodes are low (AB=00). As the supply rail is powered up both P1 and P2 conduct, and state nodes A and B begin to ramp up. As the circuit approaches its metastable point, the natural feedback causes the cell to transition to a stable state. If the cells are perfectly matched, the circuit can persist at this metastable point for hundreds of picoseconds before transitioning. In this example case, node A charges higher, and node B gets discharged through N2, allowing node A to be pulled fully high through P1, causing the cell to transition to the stable state of AB=10. When the power is removed, the charge on high node will gradually leak off, leaving the cell fully discharged until the next initialization. Note that discharge of the node A is a far slower process that the initialization. In the following sections, an explanation is given for why this initialization procedure is a function of process variation and of noise.

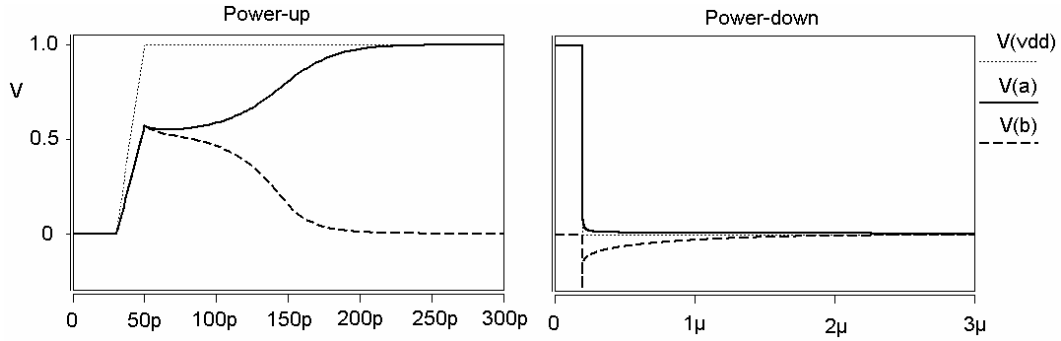

Figure 2.2: Waveforms showing power-up and power-down of state nodes of a 6T SRAM cell

2.2.1 Principles of Identity

In this section, the question of why an SRAM cell will tend to favor one state over another is addressed. Assuming that the RAM cell has symmetric design and layout, any differences between the cells

is attributable to process variation. While there are many sources of process variation, only local mismatch will have a different effect on each of the cross coupled devices, and local mismatch is primarily due to fluctuations in the concentrations of dopant atoms [35]. Prior work shows that across devices separated by only 2μm, threshold mismatch can have a standard deviation on the order of tens of millivolts [36]. The greatest threshold mismatch in MOSFETs is achieved using minimum sized devices [37], as would typically be used in SRAM.

In FERNS, threshold mismatch impacts the power up state of the cell by causing one node to begin to discharge before the other as the supply and state nodes ramp up. Even if a node begins to discharge only slightly before the opposing node, the natural gain of metastable cross coupled inverters will serve to magnify that small difference. HSPICE experiments in PTM 180nm bulk CMOS [38] using a 2:1 P:N sizing ratio are performed in order to roughly determine the sensitivity of the initial state of an SRAM cell to the threshold voltages of the devices in the cross coupled inverters. To skew the cell, node A is precharged to 1mV at the start of the simulation; in the absence of variation, this will lead to the cell initializing to the AB=10 state. To determine sensitivity to various threshold deviations, the amount of threshold voltage change on each device required to neutralize this 1mV precharge and make the cell metastable is determined (table 2.1). The finding is that the initial state of the RAM cell is approximately twice as sensitive to variation in the threshold of the NMOS devices as it is to variation in the PMOS devices. This implies that it is not merely threshold variation that is the primary source of identity in FERNS, but specifically NMOS device threshold variation. As would be expected, equivalent skew is caused by changing opposing devices of the same type in opposing directions; for example, decreasing the threshold of P1 by 2.5mV has the same impact as increasing the threshold of P2 by the same 2.5mV.

Table 2.1: A 1mV precharge on a state node is equated to a threshold voltage variation on each of the 4 relevant devices.

		Equivalent skews			
	Nominal	Var1	Var2	Var3	Var4
P1 Vth	-0.4200	-0.0025	0	0	0
P2 Vth	-0.4200	0	+0.0025	0	0
N1 Vth	0.3999	0	0	-0.0013	0
N2 Vth	0.3999	0	0	0	+0.0013
Precharge A (V)	0	0.001	0.001	0.001	0.001

2.2.2 Principles of Randomness

In SRAM cells that are not significantly skewed due to process variation, the power-up state can be determined by noise. The thermal noise model of Sarpeshkar, Delbruck and Mead is used [24] (eq 2.1). Their derivation notices that an increase in resistance will increase the magnitude of the power spectral density of noise, but will also decrease the bandwidth of the captured noise, leading to a model of thermal noise that is independent of resistance with the root-mean-squared voltage fluctuation of a single-capacitor circuit node being determined from only Boltzmann's constant, temperature, and capacitance.

$$V_n = \sqrt{\frac{K_B T}{C}} \qquad K_B = 1.38 * 10^{-23} \frac{J}{K} \tag{2.1}$$

The existence of thermal noise in semiconductors causes SRAM to capture true physical randomness. In an unpowered SRAM cell, each of the two state nodes will have noise, but the noise on each node will generally not have sufficient magnitude to turn on any transistors and the nodes will remain isolated from one another. Depending upon the state of noise conditions when the power to the cell is applied, the initial state of the cell is influenced slightly towards one state. However, this influence is generally weak, and is only sufficient to influence the initial state of a cell if that cell is constructed from well matched devices. For cells that are highly skewed towards one state or another, the noise will have no observable impact (figure 2.3)

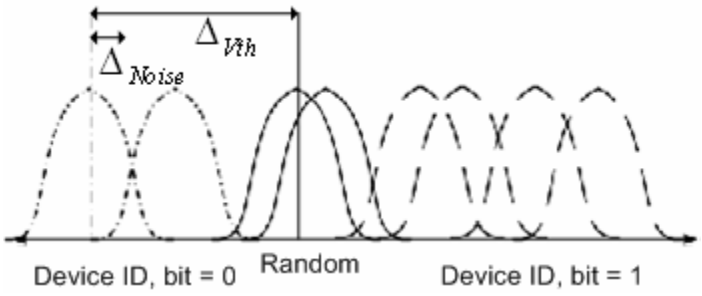

Figure 2.3: Each curve represents a single SRAM cell. The variance of each curve represents the influence of noise, while the mean represents the influence of process variation.

This ability of FERNS to capture true randomness using SRAM is very similar to the random number generator designed by Tokunaga, Blaauw and Mudge [27]. Their design uses a metastable cell that is very similar to an SRAM cell, with control circuitry used to force the cell to a highly metastable state by

precharging the circuit as needed to neutralize any skew. Once the cell is metastable, control is released and the cell stabilizes to a random state. Quality control is accomplished using a timer; if the metastability takes a long time to resolve, then some assurance is provided regarding the randomness of the output. In this design, it was found through experiment that controlling the state nodes with a resolution of 120μV was sufficient for obtaining qualified random outputs. The major difference between FERNS and this design is that we are not controlling the SRAM cells to metastability. Instead, the law of large numbers is allowed to ensure that some SRAM cells will inherently be metastable. Additionally, FERNS is not seeking to collect randomness only from cells that are perfectly metastable, but to instead collect useful randomness even from cells that are significantly skewed towards one state, but capable of initializing to the other state under strong noise conditions.

2.3 Experimental methodologies

Three platforms are used to validate the FERNS system. Virtual tags are used to allow for convenient collection of large amounts of questionably representative data. TI MSP430 chips are used to allow for collection of a moderate amount of data from a more representative low-power SRAM. Finally, a small population of WISPs is used to collect a smaller amount of data on a highly representative technology. Demonstrating FERNS across all three platforms adds confidence that FERNS can function across a spectrum of technologies.

Table 2.2: Differences between experimental platforms.

	Supply (V)	Data retention current (pA/bit)	Power source
Virtual tags	3.3	1220	external regulator
TI MSP430	3.0	49	JTAG debugger
WISP	3.0	49	Passive

2.3.1 Virtual Tags

A population of 160 virtual tags are created across eight 512kbyte ISSI 61LV25616AL [39] chips. Each virtual tag is a 256 byte segment of memory; 20 virtual tags are created on each chip, using the same addresses across all chips. Each SRAM chip is powered and read out via an Altera DE2 board [40].

Software to collect and format the data from the SRAM is written in the C language, and executed on the NIOS2 core on the DE2 board. Whilst the technology node of the ISSI SRAM chip is not known, it is specified as a high-performance chip, operating at a 3.3v supply, and has a RAM retention standby current of 5mA. The virtual tags are not intended to be a precise technology match for RFID technology, but are used as a means of collecting a large data set.

2.3.2 TI MSP430

A population of 10 TI MSP430F1232 Ultra-low-power microcontrollers [41] is also used to validate the FERNS principle. The MSP430 has an 8MHz 16-bit RISC core, 256 bytes of SRAM, 256 bytes of Flash main memory, and 8kbytes of Flash ROM. Each microcontroller is soldered onto a development board that includes a JTAG header. The size of the sample population is limited by the fact that the MSP is only available in surface mount packaging, thus requiring a development board for each microcontroller, prevent swapping multiple chips in and out from a single socket board. Communication and power is accomplished using the TI MSP-FET430 debugger [42] via IAR Embedded Workbench IDE V3.42A. The MSP430 is capable of operating on a supply ranging from 1.8 to 3.6V (a 3.0V supply was used), and has an SRAM data retention current of 0.1μA. Being an ultra-low-power design, the MSP430 SRAM is thought to be a good technology match for RFID circuits. The fact that the SRAM is embedded into a microcontroller also allows for examination of the possible impact of neighboring logic.

2.3.3 WISP

A population of 3 wirelessly-powered platforms for sensing and computation, or WISPs [43,44], is used a final experimental platform for FERNS. The WISP, produced by Intel Research Seattle, is passively powered at 915MHz in the UHF band, and transmits data in 64 bit packets according to the EPC Gen 1 specification, allowing communication with commercially available RFID readers. The WISP is built around the same MSP430F1232 microcontroller that is used on the 10 development boards. Experimenting on the WISP allows for exploration of the possible impact of passive power.

CHAPTER 3

FERNS FOR IDENTIFICATION

3.1 Introduction

FERNS extracts a usable fingerprint from the initial state of SRAM. This chapter provides experimental evidence that SRAM contains a useable identifying fingerprint, and presents algorithms for identifying tags based on their fingerprints. When considering fingerprint matching, the following terminology is adopted from human fingerprinting.

Latent Fingerprint - A fingerprint generated in RAM at initialization. A latent print represents a single data point and can be impacted by noise.

Known Fingerprint - An intentional fingerprint that is cataloged for matching against latent prints. Known prints are obtained by averaging many latent prints, to eliminate the effects of noise.

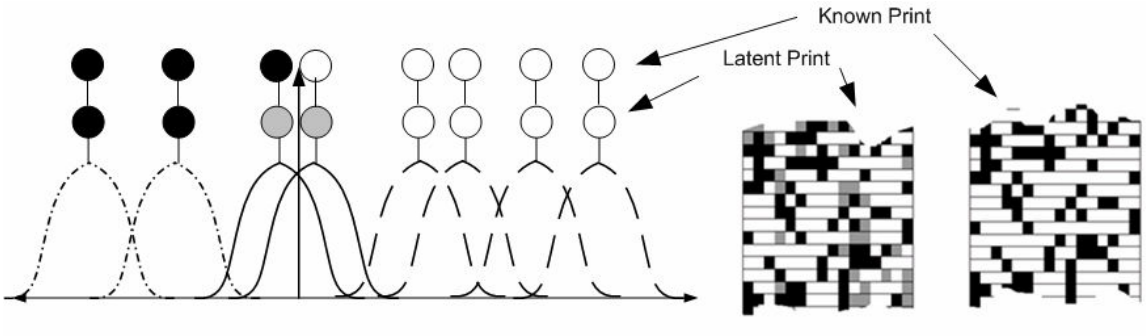

Figure 3.1: The difference between latent and known prints.

In order to reliably identify tags amongst a population, two conditions must be satisfied. A strong similarity must exist between a latent fingerprint and known fingerprint when both are generated by the same device. A lack of similarity must exist between a latent fingerprint and known fingerprints generated by all other tags.

3.2 Randomized Identification

Identification via the slightly randomized fingerprint of FERNS offers some potential security advantages. The set of possible latent prints that identify a given device can be thought of as a large ID space for the device; When the latent print contains many bits and has a realistic bit error rate (eq 3.1,3.2),

the probability of a latent print producing the same latent print on sequential sessions becomes vanishingly small. The expected number of latent prints produced before repeating the first latent print (eq 3.3), and expected number of latent prints produced before repeating any previous latent print (eq 3.4) are large enough to make repeated prints unlikely for both virtual tags and the MSP430. This is supported by experimental evidence; no virtual tag or MSP430 was found to produce the exact same latent print twice. In the WISP, using only a 64 bit identifier, both prediction and experiment show that exact duplication of latent prints is an infrequent but realistic occurrence (figure 3.2).

In light of the relative uniqueness of each latent print, a reader might prevent replay attacks by cataloging a history of the latent prints generated by a device. The condition for authenticating a tag would then be to force the tag to produce a new latent print that closely matches the known print without duplicating a previously seen latent print. Note that this would only prevent replay attacks, as an intelligent adversary could still easily generate randomized prints himself using an algorithm instead of a physical SRAM array. This is analogous to human fingerprinting, where it is not impossible for an adversary to reproduce a fingerprint; it is only impossible for him to reproduce a fingerprint using another human finger. Note that the infrequency of collisions in FERND ID is only meant to imply that replays could be detected, and is not meant to imply security in the same manner in which a collision-resistant hash function or PUF circuit might.

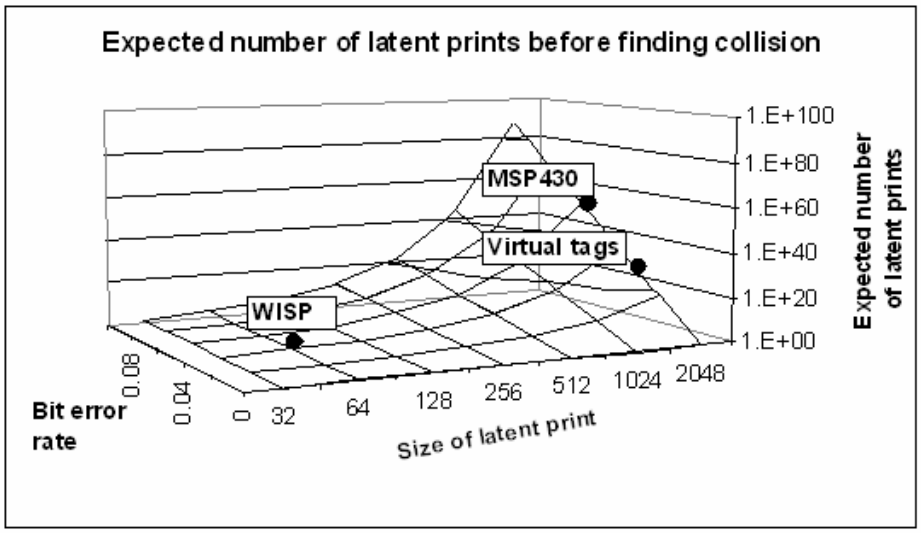

Figure 3.2: With the randomized identifier of FERNS, a particular SRAM is expected to produce many latent prints before any repetitions occur.

$$P(bit\_collision) = P(bit\_error)^2 + (1 - P(bit\_error))^2 \tag{3.1}$$

$$P(id\_collision) = P(bit\_collision)^{num\_of\_bits} \tag{3.2}$$

$$E(id\_collision) = \frac{1}{P(id\_collision)} \tag{3.3}$$

$$E(finding\_any\_collision) = \sqrt{E(id\_collision)} \tag{3.4}$$

3.3 Hamming Distance Matching

The first matching algorithm considered is a straightforward Hamming Distance based matching between a latent print of unknown origin and all known prints. Keep in mind that a known print is determined by finding the bit-wise median across many latent prints of the same tag, eliminating the impact of any normally distributed noise (eq 3.5). The Hamming Distance between a latent print and each known print is simply the number of bit positions in which the prints differ (eq 3.6). The known print that most closely matches the unknown latent print is determined to be the identity of the latent print. The term "correct match" is used to describe a latent and known print originating from the same device.

$$k(i) = (\text{as integer})\frac{1}{runs}\sum_{r=1}^{r=runs} l_r(i) \tag{3.5}$$

$$HD(l,k) = \sum_{i=1}^{i=bits} l(i) \text{ XOR } k(i) \tag{3.6}$$

3.3.1. Virtual Tag Identification

Virtual tags are questionably representative of typical RFID circuit technology, but present a potentially challenging scenario for fingerprint identification due to the large population size and the fact that virtual tags allow for testing of process corner cases. Virtual tags that occupy the same positions on different SRAM chips have correlated within-field positions, while tags from nearby locations on the same SRAM chip have correlated wafer positions (figure 3.3). If fingerprints are being caused by mask-imperfections, tags from the same within field position could show correlation. If fingerprints are being caused by wafer-scale processing, virtual tags from the same SRAM, coming from similar wafer positions,

could show a correlation. Without virtual tags or custom silicon, there would be no way determine the relative wafer positions of the circuits being compared. The observation that neither of these corner cases showed a strong correlation supports the claim that primary source of the differentiation is threshold mismatch due to random dopant concentrations. It should be noted that a majority of bits across virtual tags have a known value of 1, indicating a systematic skew in layout or design.

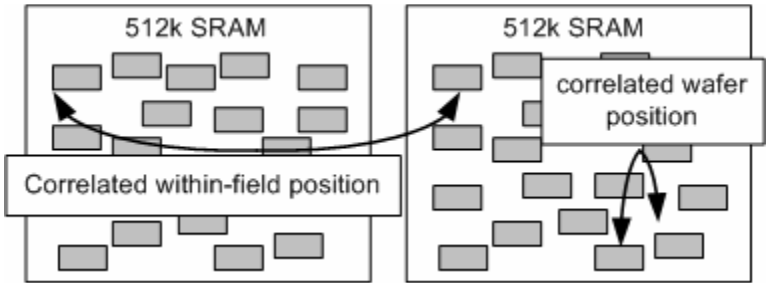

Figure 3.3: Virtual tags allow for the comparison of SRAM with correlated within-field and wafer positions.

Hamming Distance matching is applied to 5 sequentially generated latent prints from each of the 160 virtual tags. This yields 800 latent prints, each is compared to 160 known prints, for a total of 128,000 matchings between known and latent prints. All 127,200 incorrect matchings differ by at least 685 of the 2,048 bits; all 800 correct matchings differ by less than 109 bits. Given that a majority of bits tends towards the 1 state, full independent SRAM instances would show a hamming distance of 860 bits. The Hamming Distance matching results for virtual tags indicates that the tags are not significantly correlated (figure 3.4).

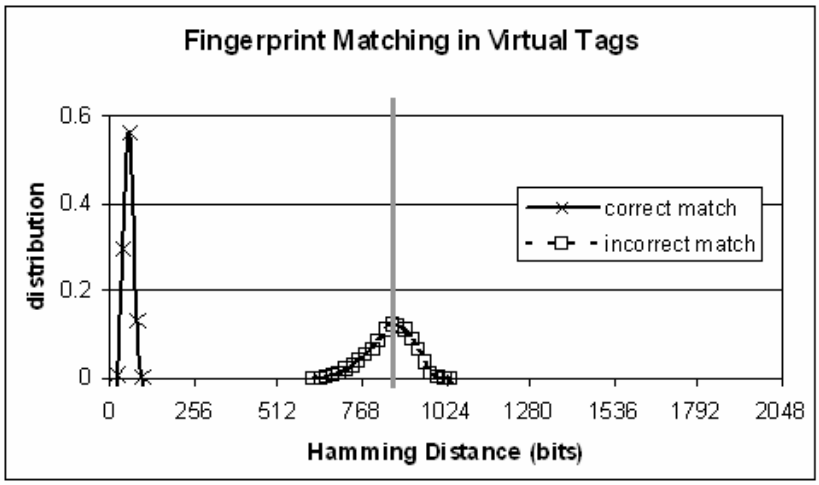

Figure 3.4: Hamming Distance matching results for virtual tags. The gray line represents the expected Hamming Distance matching that would be expected for fully independent tags.

3.3.2 MSP430 Identification

Hamming Distance based FERNS identification is also tested on the population of MSP430 microcontrollers, with communication and 3v power accomplished via the JTAG debugger. Among the population of 10 MSP430s, 30 sequential 2,048 bit latent prints are generated by each. This yields 300 latent fingerprints for comparison against 10 known fingerprints each, for a total of 3,000 possible matches. Among the 2,700 incorrect matchings, less than 10 came within 600 bits of each other. Among the 300 correct matches, only 4 differed by more than 425 bits (figure 3.5). While the overlap between incorrect matching and correct matchings indicates that identification could not be made fully reliable, this unreliability is believed to be due to an artificial correlation that is being introduced by the experiment setup, and not the circuit itself. Please see appendix A for further details on this. If the tags were completely independent, the expected Hamming distance between mismatched tags would be 992 bits. This distance is less than 1,024 bits due to the fact that 8 bytes cannot be left uninitialized, and will always agree between all tags.

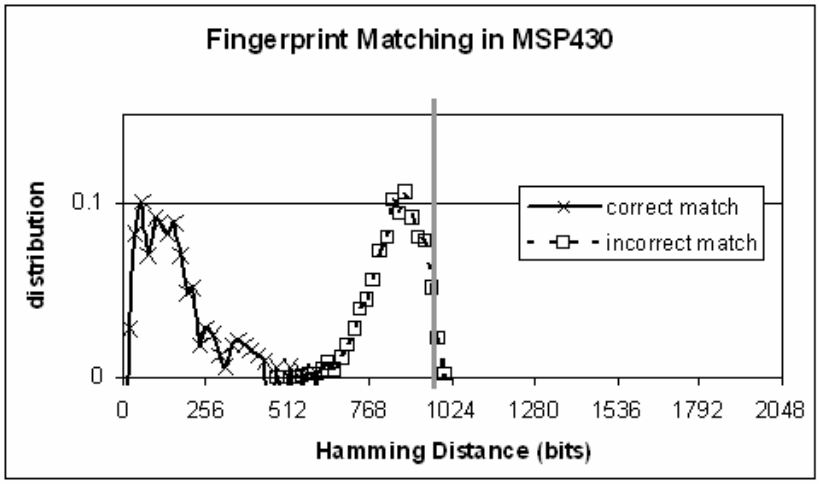

Figure 3.5: Hamming Distance based matching of MSP430. The gray line represent the expected hamming distance for mismatched uncorrelated tags.

3.3.3 WISP Identification

FERNS identification is also explored on the small population of WISPs. Because the WISP transmits 64 bit packets, comparisons are performed using 64 bit fingerprints instead of the 2,048 bit fingerprints used for the other two platforms. 5 distinct blocks of SRAM were used on each of 3 WISPs, producing 15 known fingerprints. Each segment of SRAM produced 16 sequential latent fingerprints, to

provide 3600 total possible matches. Among the 3,360 incorrect matchings, none came within 18 bits of matching. Among the 240 correct matches, one had a Hamming distance of 20 bits, and one had a distance of 16 bits (figure 3.6). This indicates that, even with an appropriate matching threshold, a small error rate is inevitable for this dataset. Note that this is using only a 64 bit print, making identification far more difficult than with the 2,048 bit prints of the other platforms. This result implies that passive power does not influence the fingerprints. In fact, as shown in appendix A, the same devices yielded more reliable fingerprints when passively powered than when actively powered through the JTAG due to a correlation seemingly induced by the JTAG itself.

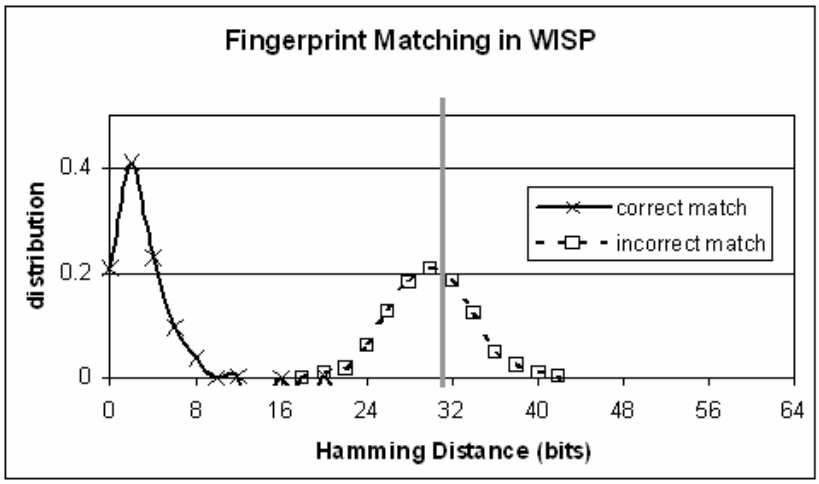

Figure 3.6: Hamming Distance based matching of WISPs.

3.4 Progressive Matching

While the Hamming distance based matching is shown to be effective, it is not scalable. A progressive matching algorithm is designed here for scalability, and adaptability. The intuition for this algorithm stems from the large hamming distance seen between mismatched pairings of latent and known prints. This large distance implies that it is inefficient to compare each bit of the latent print to the corresponding bit of each known print, since a correct match could be found using fewer bits. Note that in this progressive matching algorithm, the number of matching bits is considered instead of the differing bits; thus a high matching score represents a good match. Because the matching strength of a correct match will grow with the number of bits considered, the correct match can be identified once its match strength is

significantly higher than the match strength of any other known prints. In the case of virtual tags, it was shown in the previous section that 2,048 bits was quite sufficient to identify the matching known print corresponding to a given latent print. In fact, it is found that considering all 2,048 bits of the latent print is unnecessary, and that the tag could be identified with reasonable confidence using only 48 bits (figure 3.7).

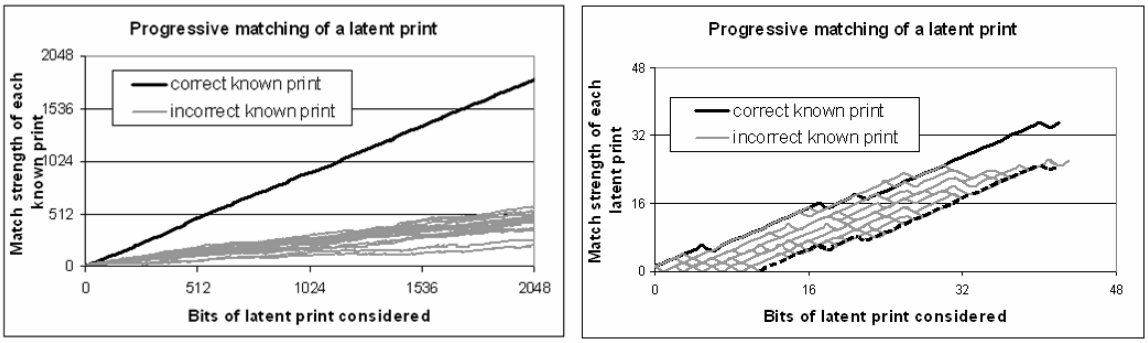

Figure 3.7: Match strength grows stronger for correct known print as more bits are considered. The plot at right shows that the correct known print can be identified without needing to consider all by pruning matches falling below a threshold.

An algorithm using progressive matching and list pruning to try to determine the correct match for a latent print is given here. In this algorithm, whenever the strongest match exceeds all other match strengths by a specified threshold it is determined to be the correct match. Any known prints which are not within this threshold are discarded from future consideration as possible matches. Thus, as more bits are considered, fewer known prints remain as possible matches, reducing the number of bit comparisons that need to be performed for future bits (figure 3.8).

Progressive Matching algorithm:

Initialize a list where each item represents one known print that is a possible match
While length of list > 1
    Foreach known print K in list
        match strength$_K$ += (current bit$_K$) XNOR (current latent bit)
    If (maximum match strength – match strength$_K$) > thresh
        Remove K from list
Update maximum match strength
Advance to next bit of latent print

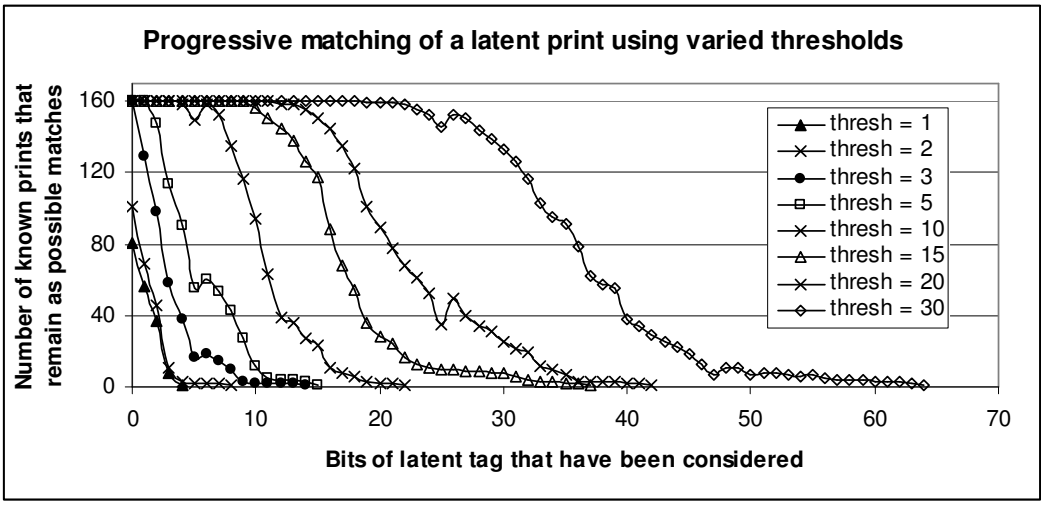

Figure 3.8: The pruning of incorrect matches in progressive matching algorithm.

The threshold for the progressive matching needs to be carefully chosen according to a tradeoff in accuracy versus performance. A small threshold will aggressively prune the list of possible matches, but may prune out the correct known print. As an example, note that the correct match was not the strongest match through the first 24 bits in figure 3.7; a small threshold could have thus resulted in an incorrect match. On the other hand, a large threshold will be more likely to produce the correct match but will not prune known prints as aggressively and thus requires more bitwise comparisons before choosing a match (table 3.1).

Table 3.1 Comparison of efficiency and accuracy of thresholds in progressive matching algorithm.

threshold used	bits of latent print considered	total bits of known prints considered	success rate
1	8.65	438.1	63.50%
2	9.85	504.8	73.75%
3	15.38	957.3	95.63%
4	16.26	1,030.0	97.13%
5	20.02	1,486.0	99.63%
6	20.56	1,577.1	99.75%
7	24.33	2,047.4	99.88%
8	24.78	2,151.3	99.88%
9	28.28	2,623.7	100.00%
10	28.79	2,728.7	100.00%
15	39.30	4,303.5	100.00%
20	47.03	5,490.4	100.00%
Hamming	2048.00	327,680.00	100.00%

The progressive matching algorithm and Hamming Distance matching represent opposing approaches. In the Hamming Distance matching algorithm the number of bit comparisons is maximum, but the overhead of comparing matching strengths is low as the Hamming Distance needs only to be considered once per known print, at the end of the algorithm. The progressive matching algorithm is designed to minimize the total number of bits that are considered, but has an increased overhead in terms of comparing matching strengths; after each single bit comparison, the matching strength of the relevant known print must be checked against the threshold and the maximum matching strength must be updated. It is likely that this overhead would make the progressive matching algorithm prohibitively expensive. An optimal matching solution depends on the costs assigned to the various operations, but would likely be a compromise between the two approaches. An example of such a compromise is a progressive matching algorithm that only does the pruning after each byte of latent print, instead of each bit. This would retain the adaptive nature of the progressive matching, but would reduce the overhead.

CHAPTER 4

FERNS FOR RANDOM NUMBER GENERATION

4.1 Introduction

FERNS captures physically random noise in the initial state of SRAM cells that are constructed from well matched devices. In essence, the well matched cells of SRAM are tiny 6 transistor random bit generators. These random bits are inherently stored in SRAM, and a universal hash function is used to extract a smaller number of highly random bits from the fingerprint. While most random bit generators seek to create bits that are highly random, FERNS takes a different approach by gathering randomness from all bits that are not fully deterministic. In this chapter, only virtual tags are used as an experiment platform; this is done because the virtual tags are the least random of the three platforms, and thus present the most challenging case for random number generation.

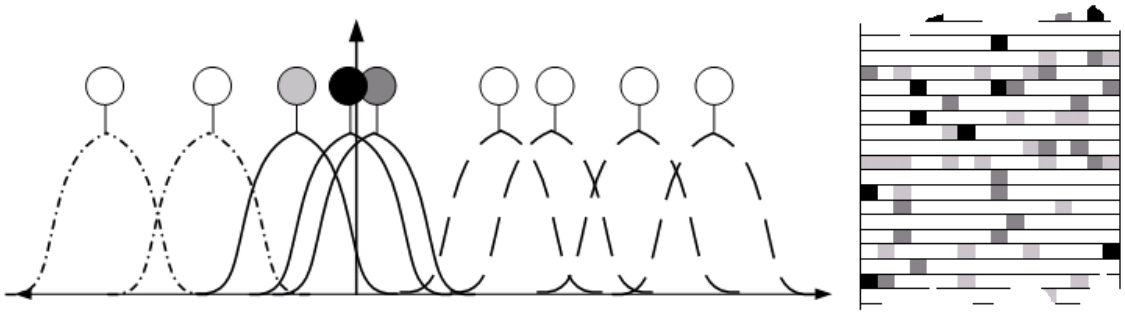

Figure 4.1: FERNS relies on law of large numbers to collect scattered entropy from well matched cells.

4.2 Potential security features of using FERNS for TRNG

FERNS is an unusual TRNG with interesting security implications, and is potentially resistant to attack for several reasons. Because the well matched devices are randomly scattered according to dopant concentrations, the entropy generation is unpredictably scattered throughout the SRAM state. If the application for FERNS could ensure that the initial SRAM state could be kept secret, then even an attacker who had designed the circuit herself would not know the location of the random bits. If an attacker did identify the random bits, the densely packed cells might make it difficult for her to actively control some bits without also influencing others. Further, the distributed parallelism of RAM provides a natural

resiliency against attacks. FERNS TRNG also has some potential security weaknesses that must be addressed, including the possibility of an attacker reducing entropy by physically cooling down a chip to reduce the magnitude of thermal noise; this is discussed later in the chapter.

4.3 Metrics for Quantifying Randomness of initial RAM State

The Shannon information content of an outcome of a random variable is the amount of information that is learned from the output. If a fair coin is flipped, the outcome of the flip can take two states with equal probability, thus the outcome reveals a bit of information. When rolling a fair 6 sided die, the outcome can take 6 states, and thus the outcome reveals more information (2.585 bits). If the outcomes are not weighted evenly a less common outcome yields more information; intuitively, this is analogous to the fact that learning that a person of unknown identity is 7ft tall yields more information about their identity than does learning that are 6ft tall. This notion of information content of an outcome is formalized below (eq 4.1) from [45]

$$h(x) = \log_2 \frac{1}{P(x)} \tag{4.1}$$

The entropy, or uncertainty, of the random variable itself is typically more useful than the information content of a particular outcome. The entropy of an ensemble of outcomes is the expected value of the Shannon information content of an outcome from that ensemble. The entropy of a fair coin flip is a bit. However, if the coin was weighted to produce heads 90% of the time, the entropy of the coin flip would decrease to 0.47 bits. The intuition for the decrease in entropy is that the outcome of the flip can be predicted with some certainty even before the coin is flipped. Entropy is formalized below (eq 4.2) from [45]

$$H(X) = \sum_i P(x_i) \log_2 \frac{1}{P(x_i)} \tag{4.2}$$

In this work, a more useful definition is that of **min entropy**. Min entropy refers to the guessing probability associated with a random variable. The guessing probability is the likelihood that a random variable will produce its most probable outcome (eq 4.3). Min entropy is a measure of the entropy contained in that most probable outcome (eq 4.4). The usefulness of min entropy can be understood by considering the following two systems. The first contains 8 independent bits where each can be guessed

with probability 0.8, and the second contains 4 bits where each can be guessed with probability 0.64. For each system, an adversary would clearly have the same chance of guessing the outcome of the system. This is properly reflected in the fact that each system would have the same min-entropy of 2.575 bits (equivalent to 2.575 perfectly random bits). However, the 8 bit system has greater information entropy than the 4 bit system (5.775 vs. 3.771 bits). This example of why information entropy does not accurately describe the difficulty of guessing an outcome serves to indicate why it is min entropy and not information entropy that is appropriate in random number generation. Informally, the goal in random number generation is to obtain state that cannot be guessed by an adversary.

$$\gamma(X) = \max_i P(x_i) \tag{4.3}$$

$$H_\infty(X) = \log_2 \frac{1}{\gamma(X)} \tag{4.4}$$

4.3.1 Entropy of Initial SRAM State

In order to use FERNS for random number generation, the min entropy of the SRAM must be quantified. This requires a larger dataset than for verifying the ID properties of FERNS; the dataset used for quantifying min-entropy is 100 samples of 524,288 bits (256 blocks of 2048 bits each) from a single virtual tag SRAM chip. Recall that quantifying min entropy of a random variable requires finding the most likely outcome of that random variable. This leads to a tradeoff in how the random variable is defined. If the size of the random variable is a single bit, correlations between neighboring bits can be missed, as is demonstrated here using the example of table 4.1a. Considering each bit to be a single random variable would indicate that this dataset contains around 3 bits of min-entropy. Using a larger random variable size would correctly determine that this dataset contains closer to 1 bit of min-entropy by noting the correlation between neighboring bits. At the other end of the spectrum, using a random variable size that is too large for the number of samples in the dataset can also lead to an erroneous quantification of entropy, as in table 4.1b. In this case, considering the size of each random variable to be a single bit would lead to a correct entropy determination. However, if the size of each random variable was defined to be 5 bits wide, the most likely outcome would occur once in 5 trials, while a truly random outcome should occur only once in 32 samples. In this case, the min entropy would be artificially governed by the size of the random variable.

Table 4.1: Examples referred to in discussion of min-entropy.

a:

sample 0:	0	0	0
sample 1:	1	1	1
sample 2:	0	0	0
sample 3:	0	0	0
sample 4:	1	1	1

b:

sample 0:	0	0	1	1	0
sample 1:	1	1	0	0	1
sample 2:	0	1	1	0	1
sample 3:	1	0	0	1	1
sample 4:	1	0	0	1	0

Due to the concerns about how to quantify the min entropy of the dataset, random variables of differing sizes are considered, and the results are discussed here (figure 4.2). As the size of the random variable is increased from 1 bit to 16 bits, no significant drop in average entropy per bit is seen, indicating that neighboring bits are not highly correlated. As the size of random variable reaches 32 and 64 bits, the min entropy per bit is shown to drop. This is an artifact of the size of the random variable. 100 samples are only enough to accurately measure around 4 or 5 bits of entropy per random variable. With each random variable encompassing 64 bits, the random variables can each contain more than 4 bits of entropy, and the entropy measurement is thereafter artificially limited by the number of samples. When the size of the random variable is increased to 256 bits, this trend becomes very clear; the min entropy per bit drops off, and the min entropy per random variable saturates at $\log_2(100)$. This saturation indicates that the most likely outcome for each random variable occurs with probability 0.01. But even a perfectly random variable of an infinite number of bits would show a most likely outcome probability of 0.01 if only sampled 100 times. The observation that the calculated entropy per bit only begins to drop off when approaching this saturation point gives confidence to a determination of 0.05 bits of min entropy per bit of SRAM at room temperature. For further information regarding min-entropy, see [46].

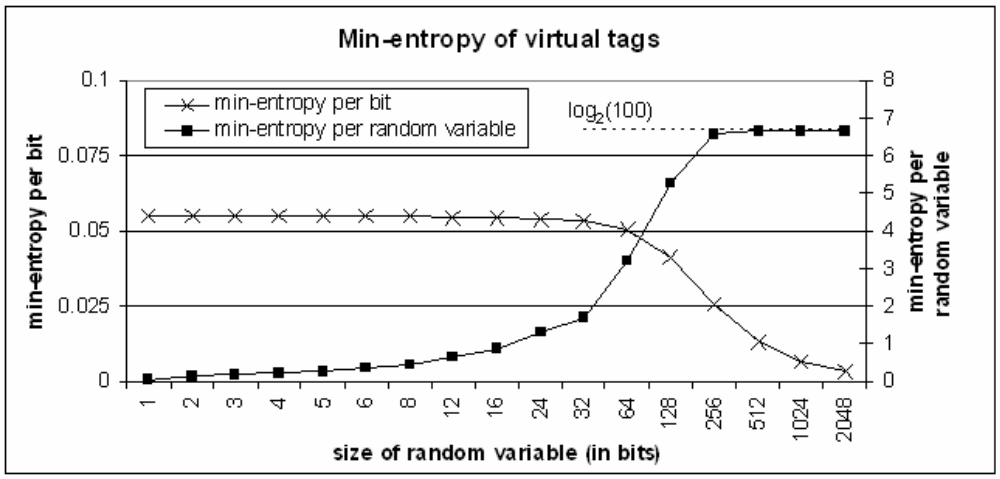

Figure 4.2: Points A and B correspond to the tradeoff described above. Point A considers each bit to be a random variable, Point B considered each 2048 outcome to be a random variable.

4.3.2 Analytical Model of Impact of Noise versus Process Variation

A model of the relative impacts of process variation and noise is fit to a dataset consisting of 100 initializations of 524,288 bits of virtual tag at room temperature. There are too many degrees of freedom to create a fully deterministic model, but an analytical model of the relative impacts of process variation and noise on the initial state of an SRAM cell is demonstrated. The influence of both process variation and noise are quantified in terms of "skew". A negative skew represents a cell initializing to the 0 state, and positive skews represent the 1 state. While both process variation and noise are modeled as random variables, each has a different scope; Process variation is assumed to be random at manufacture time and then fixed over the life of the circuit, whereas noise is random each time the circuit is powered up, and depends on temperature.

The first step in creating an analytical model is to determine the skew caused by noise. The root-mean-squared magnitude of thermal noise, introduced in chapter 2, is equivalent to standard deviation of thermal noise. The variance is thus simply the square of the root-mean-squared noise (eq 4.5). In determining the influence of noise on the outcome of the cell, it is assumed that the relevant quantity is not the absolute noise magnitude, but instead the noise differential between the two state nodes (eq 4.6). Temperature is set to 295 K to model the ambient environment where the experimental dataset was collected. Node capacitance is rather arbitrarily set at 0.1fF as an estimate of what might be seen in a minimum size SRAM cell.

$$\sigma_{NOISE}^2 = \frac{K_B T}{C} \tag{4.5}$$

$$X_{NOISE} \sim N(0, \sigma_{NOISE}^2) \qquad X_{DIFFERENIAL} \sim N(0, 2\sigma_{NOISE}^2) \tag{4.6}$$

Next, the impact of process variation is considered. Like noise, process variation is also described as a normally distributed random variable. The first pass at approximating the influence of process variation comes from the observation that the dataset consists of 25.4% bits that are more likely to initialize to 0, and 74.6% bits that are more likely to initialize to 1. This indicates that the normally distributed process variation model should have 25.4% of its skew distribution be negative, and 74.6% be positive. While this information is not sufficient to determine both the mean and standard deviation of process

variation, it is sufficient to determine that the ratio of mean to standard deviation should be 0.661; any normal distribution adhering to this ratio will produce the desired distribution between positive and negative skew (figure 4.3).

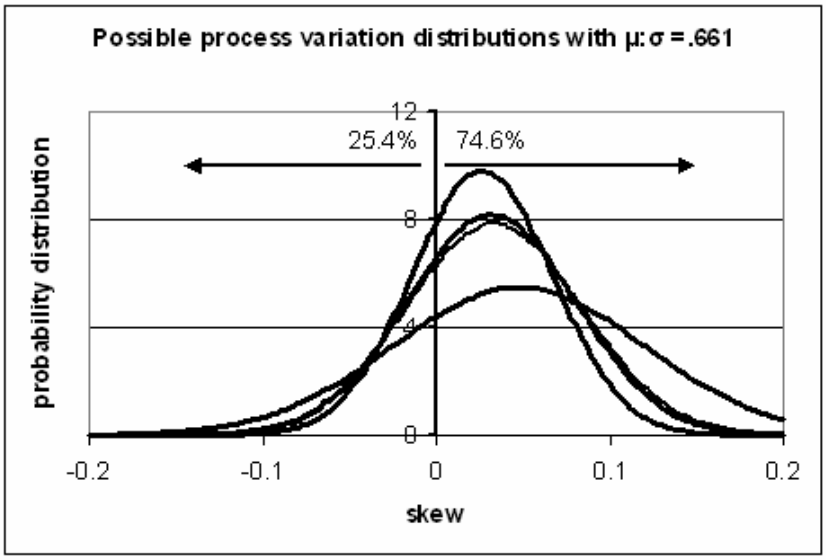

Figure 4.3 Family of process variation distributions that would meet the requirement of producing 74.6% bits skewed towards 1.

$$X_{PV} \sim N(s\mu_{PV}, (s\sigma_{PV})^2) \qquad \frac{\mu_{PV}}{\sigma_{PV}} = 0.661 \tag{4.7}$$

The scaling factor "s" is determined according to the distribution of bitwise outcome probabilities observed in the dataset. If this scaling factor "s" is set too large relative to noise, then too few bits will be predicted to be random by the model (process variation will dominate noise); if the scale "s" is set to be too small relative to noise, then too many bits will be predicted to be random by the model. Based on this principle, "s" is determined iteratively; each time a guess for "s" is made, the distribution of outcome probabilities is calculated, and the error between the prediction and observation is determined. The proper value for "s" is determined to be that which minimizes the error between model and experiment, according to the following 3 step process.

(1) Convert "s" into a histogram of representative skews.

This step transforms the continuous probability distribution function representing process variation into a discrete histogram of discrete skews. To avoid undesirable integer effects, the histogram is

created using a small bin size of 0.00025, allowing the process variation skews to be grouped into 2000 bins. The fraction of cells expected to have skew "I" is denoted by F(I). Figure 4.4 demonstrates this process, but using many fewer bins, for clarity.

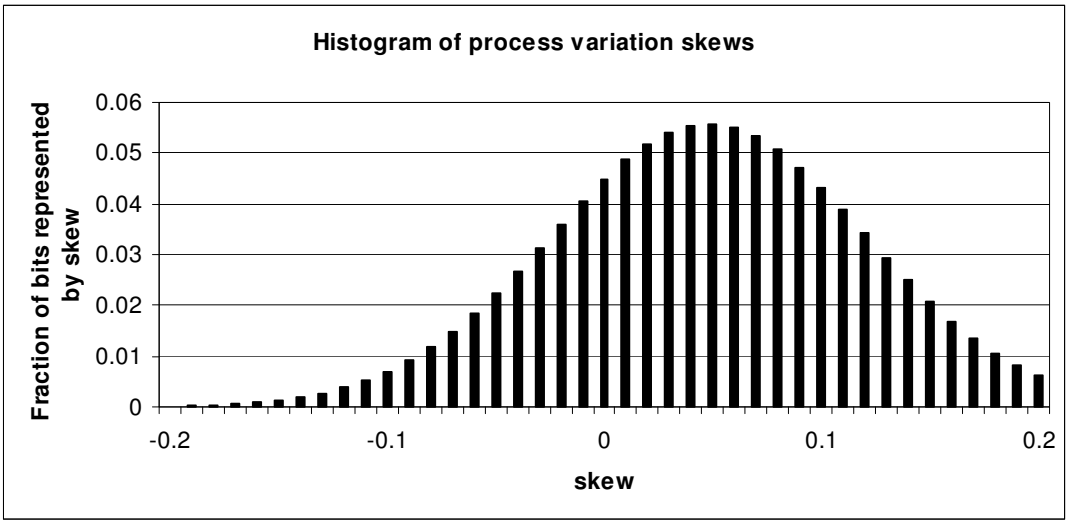

Figure 4.4 Histogram created from a process variation mode with s = 0.048.

(2) Find expected distribution of outcomes for "s."

Now that the fraction of bits represented by each skew is known, each skew "I" can be considered individually. The possible skews of a bit with process skew "I" can be described as normally distributed with a mean of I and a standard deviation caused by noise (eq 4.8). Recalling that a positive skew represents an outcome of 1, the likelihood of a bit initializing to 1 on a given trial is determined by integrating the distribution of skews over all positive values (eq 4.9). Based on this probability, the fraction of 100 sample trials in which this bit would produce "x" 1-outcomes (ie, the bit initializes to 1 "x" out of 100 trials) is determined (eq 4.10). The total share of bits that are expected to produce "x" 1-outcomes out of 100 trials according to this process variation model is determined by considering the likelihood of each skew occurring and the conditional probability that a bit with this skew would produce "x" 1-outcomes (eq 4.11). This expected distribution of bit probabilities can be compared against the observed dataset in order to determine the quality of this process variation model.

$$X_I \sim N(I, \sigma^2_{NOISE}) \tag{4.8}$$

$$p_I = \int_0^{\infty} \frac{1}{\sigma_{NOISE}\sqrt{2\pi}} \exp\left(-\frac{(x-I)^2}{2\sigma_{NOISE}^2}\right) \tag{4.9}$$

$$P_I(x) = \binom{100}{x} {p_I}^x (1-p_I)^{100-x} \tag{4.10}$$

$$E(x) = \sum_I F(I) P_I(x) \tag{4.11}$$

(3) Evaluating the error associated with choice of "s."

The accuracy of the process variation scale "s" is determined by comparing its predicted distribution of outcomes against the observed distribution of outcomes. For each possible number of 1-outcomes that could be produced, the fraction of bits predicted to produce this outcome is compared against the fraction of bits that are observed to produce this outcome (eq 4.12). The error is the accumulated square of the differences between observed and predicted values.

$$Err(s) = \sum_{x=0}^{100} \left(E(x) - Obs(x)\right)^2 \tag{4.12}$$

After many scale factors were tried, the value of "s" that produced the least error was found to s = .048, which is reflected in the constants of equation 4.13. Figure 4.5 shows this modeled distribution plotted over experiment data, with a log scale being used to allow both small and large probabilities to be seen.

$$X_{DIFFERENIAL} \sim N(0,.009011) \qquad X_{PV} \sim N(0.048, 0.072576^2) \tag{4.13}$$

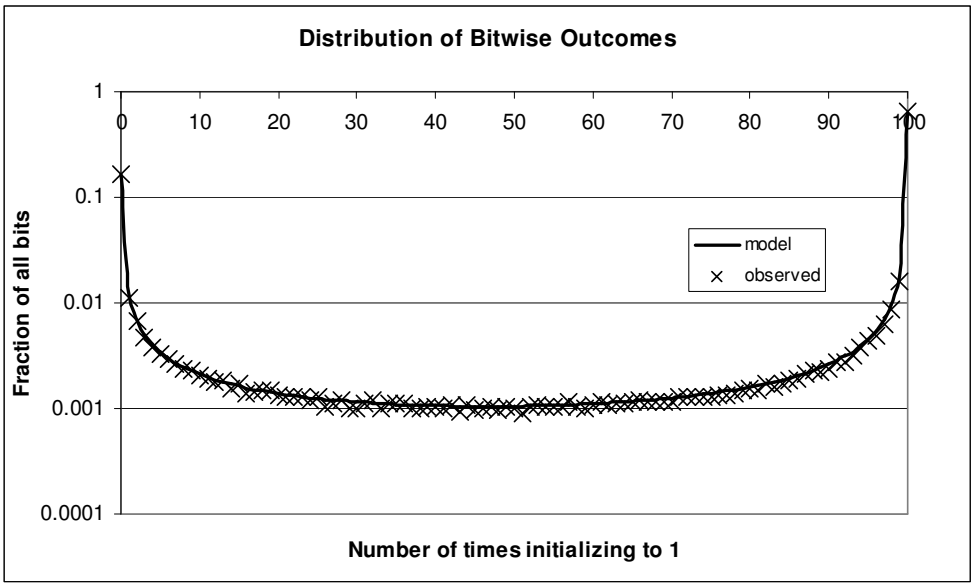

Figure 4.5 Comparison between predicted outcome distributions and experimental results

4.3.3 Demonstrating True Randomness

In chapter two, the claim is made that the primary source of randomness in FERNS is truly random thermal noise, and this was the only noise source included in our model. It was suggested that other forms of noise are likely to be common mode, and thus would not have significant impact. This distinction is important, because any noise that is not truly random could potentially be controlled by a clever adversary. Here, the claim of true randomness is explored through the use of both the analytical model developed in the previous section, and testing at varied temperatures. Testing was performed at 125F elevated over a stove and tested at 35F in a refrigerator. The standard deviation of thermal noise in the model of section 4.2.2 is recalculated for each temperature (table 4.2), and used to make predictions for the average min-entropy per bit at each temperature (figure 4.6).

Table 4.2 Predicted RMS noise voltages at each temperature used in experiment.

Temperature	RMS Voltage
35F	0.00616
70F	0.00637
125F	0.00670

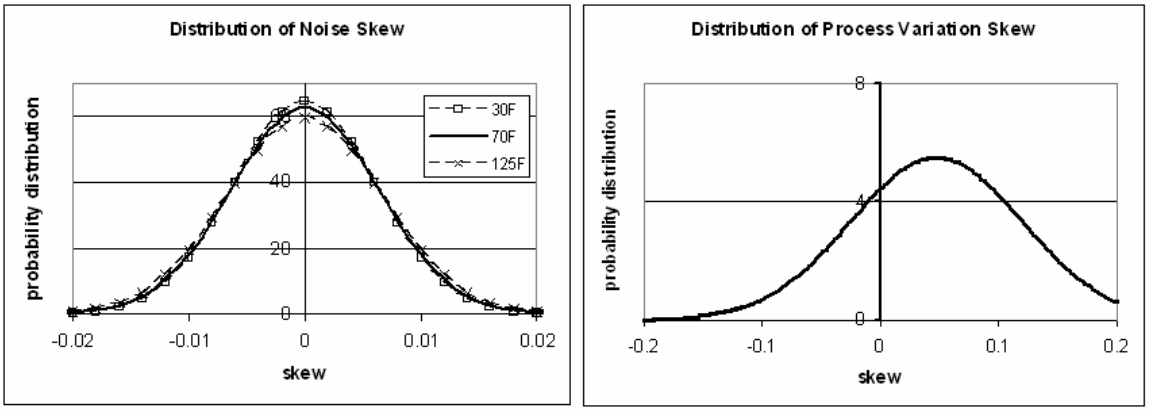

Figure 4.6: Increase in temperature increases the relative skew contribution of noise slightly, while not changing the model of process variation.

The model and prediction show moderate agreement (figure 4.7). As only 10 samples at each temperature were made, bitwise probabilities can show strong integer effects, but it is yet unclear exactly how this would impact the data. To show a more informative relation between model and experiment, more temperature points and more precise temperature control should be used. This would allow for refinement of the model based on experiment, and will likely be a topic of future work.

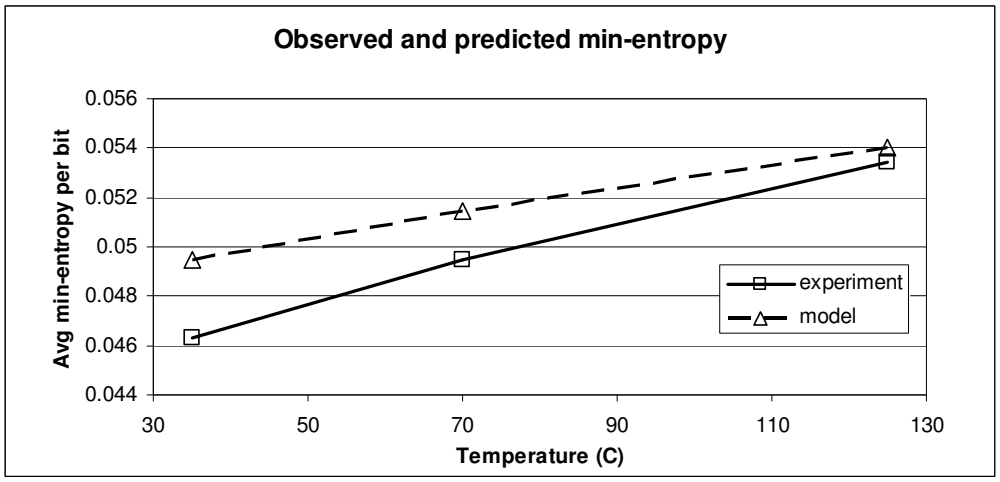

Figure 4.7: Increase in min-entropy with temperature, experiment and model.

4.4 Entropy Extraction

Now that entropy is quantified, entropy extraction is considered. Approaches to entropy extraction are the subject of much research. The most common type of randomness extraction is to simply run the weakly random source through a universal hash function [47,48], or a block cipher in feedback mode [49].

In this work, entropy extraction is performed by running the SRAM state through a universal hash function. The chosen hashing function is the PH hashing function of Yuksel, Kaps, and Sunar [50], which is stated to be 2^{-w} almost universal. PH is designed for low gate count and power, using 15.5 μW at 500kHz in 0.13μ TSMC CMOS and requiring only 557 gates. With the operations being performed over GF(2), the low gate count is due to the fact that addition and multiplication reduces to a series of shifts and XORs. In the definition of PH given below (eq 4-14), P_w is defined as the set of polynomials over GF(2) of degree less than w.

$$M = (m_1, ..., m_n) \qquad K = (k_1, ..., k_n) \qquad m_i, k_i \in P_w$$

$$PH_K(M) = \sum_{i=1}^{\frac{n}{2}} (m_{2i-1} + k_{2i-1})(m_{2i} + k_{2i}) \tag{4-14}$$

Although PH was chosen for low hardware cost, it is implemented in software for demonstration purposes in this work. Both the key and the message come directly from the 2048 bits of initial SRAM state, with 1024 bits serving as each (eq 4-15). With PH being 2^{-w} almost universal, the probability of collision is expected to be less than 2^{-64}.

$$M = (m_1, ..., m_{16}) \qquad K = (k_1, ..., k_{16}) \qquad m_i, k_i \in P_{64}$$

$$PH_K(M) = \sum_{i=1}^{8} (m_{2i-1} + k_{2i-1})(m_{2i} + k_{2i}) \tag{4-15}$$

The approximate entropy test from the NIST suite is used to evaluate the quality of the random numbers produced by the extractor [51]. For the sake of comparison, both the raw SRAM state that is input to the universal hashing function and the extracted random output are tested. Based on testing 800 blocks of 128 bits each, the raw bits show a non-uniform distribution of p-values, indicating that they can be statistically distinguished from random bits. The random bits coming out of the hash function pass the NIST approximate entropy test, indicating that random numbers capable of passing basic statistical tests can be extracted from the initial state of SRAM by use of an entropy extracting code (table 4.3).

Table 4.3 Output from NIST approximate entropy test.

dataset	*C1*	*C2*	*C3*	*C4*	*C5*	*C6*	*C7*	*C8*	*C9*	*C10*	*PVAL*	*PROP*
Raw	24523	244	76	75	34	19	10	14	4	1	0.0000	.0706
Extracted	2661	2614	2557	2590	2499	2526	2589	2570	2407	2487	0.0282	.9889

CHAPTER 5

CONCLUSIONS

5.1 Introduction

In the preceding chapters, FERNS has been demonstrated as a method for both ID and RNG. In this chapter, the viability of FERNS is considered through comparisons to existing similar works, consideration of potential weaknesses, and suggested future improvements in both design and analysis.

5.2 Comparison to RNG of Tokunaga, Blaauw, and Mudge

The TRNG aspect of FERNS is compared to the very similar design of Tokunaga, Blaauw and Mudge [27]. Recall that this design uses a metastable cell much like an SRAM cell. In their design, the cells are forced to metastability, and the stable state is determined by noise. The quality of the random numbers is judged by monitoring the amount of time that is required for the metastability to be resolved. FERNS takes the slightly different approach of not creating any specific metastability, but ensuring that some noise is captured by extracting over the entire SRAM state.

A comparison of the areas required for each approach is given. The control circuitry to generate the metastability and judge the quality of the randomness dominate the total area of Tokunaga's circuit. The area of FERNS is estimated according to the following methodologies. The size of the PH universal hashing function is estimated based on the stated cell count of 557[50] and the size of XOR standard cell from a library [52]. Because the library is 0.25µ and the design of Tokunaga uses 0.13µ, a fair comparison is created by scaling down the cell area down according to the square of the technology node. The comparison shows that area required to implement the RNG functionality of FERNS is comparable to the area required by the existing design (table 5.1).

Table 5.1: Comparison of implementation area for RNG designs.

Tokunaga, Blaauw, Mudge	Area (μm^2)
Metastable Module	6,000
Control	29,900
Total	35,900

FERNS	Area (μm^2)
256B SRAM	18,800
PH hashing function	7,400
Total	26,200

A comparison of the power required for each is given. To attempt a fair comparison, the two designs are compared under the constraint of generating 128 bits per second. Tokunaga's TRNG produces 200kb/s on 1mW power. The assumption is made that this can be scaled down to 128 b/s by simply generating 128 bits and then turning off the RNG so that no further power is needed; under this assumption, 128 b/s could be generated using only 0.640μW of average power. The PH hashing function used in FERNS is stated to use 15.5uW at 500 kHz using 0.13μ in TSMC CMOS. Although the amount of computation being performed per cycle is not entirely clear in the literature, it is believed that 2,048 cycles of the universal hashing algorithm would be required to hash the 2,048 bit SRAM state to the 128 bit random output. This computation would require only 4ms, so the assumption is made that the hashing circuit is turned off 99.6% of the time, for an average power consumption of only 0.06μW. The SRAM itself requires only .3μW of power for data retention; assuming that 10 times as much power is required for reading SRAM and that SRAM is only kept alive for the required 4ms and turned off for the other 99.6% of the time, an average power of 0.012μW is required for the SRAM. Thus, FERNS would require only 0.072μW of power, compared with 0.64μW for the reference design. This rough estimate shows that FERNS requires an order of magnitude less power while also being comparable in size. While this comparison is fairly rough, one suspected reason for the lower power consumption is that the design of Tokunaga must necessarily switch with 50ps period in order to accurately count the metastability time. In FERNS, all of the computation can be slowed down without loss of functionality, allowing aggressive min energy operating points to be used. Note that this comparison is in the special case where only 128 bits of randomness are needed, as FERNS is unable to produce larger numbers of random bits from a 2,048 bit SRAM.

5.3 Comparison to Chip ID Circuit of Su, Holleman and Otis

FERNS is next compared to the two layouts of the chip ID circuit of Su, Holleman, and Otis [32]. While an SRAM cell requires only 6 transistors, the cross-coupled NOR design of Su, Holleman and Otis requires 10 transistors for the symmetric layout, and 20 transistors for the common centroid layout. Additionally, it should be noted that great effort is put into minimizing SRAM cell size, while less efficient analog layout techniques are required by Su, Holleman and Otis in order to optimize the ID functionality. Thus, one can reasonably assume that ID generating SRAM cells used in FERNS are 50% as large as the

symmetric layout and 25% as large as the common centroid layout. Despite the much smaller and easily reusable SRAM ID cells of FERNS, the bits are only a few times less reliable. One caveat regarding the bit error rates stated below is that the work of Su, Holleman and Otis does not give a bit error rate, but instead gives a percentage of unstable bits; the bit error rate is estimated here to be half of the percentage of unstable bits, as unstable bits must produce some correct value at least half of the time. A key difference between these two works is the usage model. In FERNS, the ID can only be generated when power is first applied; in this prior work the ID can be generated at any time.

Table 5.2: Comparing FERNS ID to custom chip ID

	Su, Holleman, Otis		FERNS		
	Symmetric layout	**Common centroid layout**	**Virtual tags**	**MSP**	**WISP**
bit error rate	0.015	0.019	0.028	0.080	0.048
bitwise hamming distance	0.505	0.501	0.417	0.414	0.469

5.4 Comparison to FPGA Intrinsic PUF

Philip's Research Laboratories has designed a system for FPGA intrinsic PUFs that is similar to the work of this thesis [53]. This work will be published in the September 2007, but an advanced copy was obtained from the authors; it was developed simultaneously to, and independently of, this thesis. To create an FPGA intrinsic PUF, error correcting codes are used to extract a reliable secret from the initial state of an SRAM block. This secret is then used with a hashing function to produce PUF-like functionality. Like FERNS, this work on FPGA intrinsic PUFs recognizes that initial SRAM state has identifying properties. Unlike FERNS, no constructive use is made of the thermal noise-induced randomness. The FPGA intrinsic PUF is an expensive design, and is unsuitable for RFID tags as it requires 512 kbits of SRAM to generate 110 challenge response pairs.

5.5 Open Questions

This thesis is meant to be an initial study on the feasibility of a FERNS system. With FERNS being novel, many questions can be raised about possible weaknesses or potential oversights. In this section, some such doubts are presented and addressed. Best guesses and unsubstantiated predictions are made where possible, such that the substantiation or refutation of the predictions can provide a good starting point for future research.

5.5.1 Threshold Shift due to NBTI

One topic that must be explored further is the shift of threshold voltages that can occur over time. A primary cause of this in PMOS devices is Negative Bias Temperature Instability (NBTI), which describes interface trapped charges and holes trapped in gate oxide [54]. Several aspects of the RFID usage model suggest that NBTI might not be a great concern. The shifts caused by NBTI are known to recover when stress conditions are removed [55]; passive RFID circuits are typically unpowered before use meaning that they could have adequate time to recover from any shifts. Secondly, NBTI has cumulative effects based on the total time that the stress conditions are applied [56]; as RFID circuits are typically unpowered most of the time, cumulative effects may not be a great concern. Thirdly, NBTI has an increasing impact at smaller feature sizes in advanced technology [57], while RFID circuits use older larger technologies.

It is well know that the effects of NBTI result in delay degradation in logic circuits [58], but the impact on FERNS is less clear. It is believed that NBTI will actually act to destroy identity. Consider the following situation, extending the example of chapter 2.1. If A cell stabilizes to the state A=1 B=0, PMOS device P1 will be under stress, and will have its threshold voltage increased. This will make it more difficult for P1 to turn on next time, favoring initialization in the A=0 B=1 state, opposite the state that was "burned in". It is thus predicted that NBTI burn in will not reinforce identity, but will work against identity, increasing randomness. This is supported by two runs of a preliminary experiment where a state of all 0s was burned into SRAM over the course of a few days. Afterwards, the power was turned off for twenty seconds, believed to be long enough to lose state but not recover from NBTI, and then turned back on. In both cases the initial state was found to contain significantly more ones than usual, opposite the 0s that were burned in, and in accordance with the predictions made above. Two runs cannot be taken proof, and more experimentation is needed to confirm this finding.

5.5.2 Side Channel Attacks

Side channel attacks to either the ID or the RNG functionality of FERNS must also be considered. One potential side channel attack would be for an adversary to reduce the temperature of the device and

thus decrease the magnitude of thermal noise and reduce its entropy; as was demonstrated in chapter 4.3 of this thesis.

A second class of potential side channel attacks are related to data remanence, or the ability of SRAM to hold its value after power is removed. Such an attack could be carried out by either using data remanence to influence the initial state of an SRAM, using data remanence to predict the state, or using data remanence to reinforce the most likely outcome of each bit in attempt to destroy randomness. Kuhn and Anderson showed that data stored persistently in an SRAM from the late 1980s had "burned in" to the memory, and could be read out intact later, after the chip had been powered off, with only 5-10% error rate [59]. This is also noted by Gutman, who indicates that this occurrence is specific to older technologies [60]. Our research shows no indications of long term data remanence, except for the aforementioned NBTI effects which appear to cause SRAM to tend to initialize to the inverse of its last state. Our finding on the lack of long term remanence is supported by the findings of Skorobogatov, who tested 8 SRAM chips from the 1990s and showed that none retained state for longer than a few seconds [61]. It is predicted that when both data remanence and NBTI are considered, remanence should cause a correlation if the power down time is not long. However, if the chip has been powered on for a long time before power down, NBTI should cause an anti-correlation once data remanence has faded (figure 5.1). A third type of potential side channel attack which has not yet been explored is influence by exposition to an electromagnetic field.

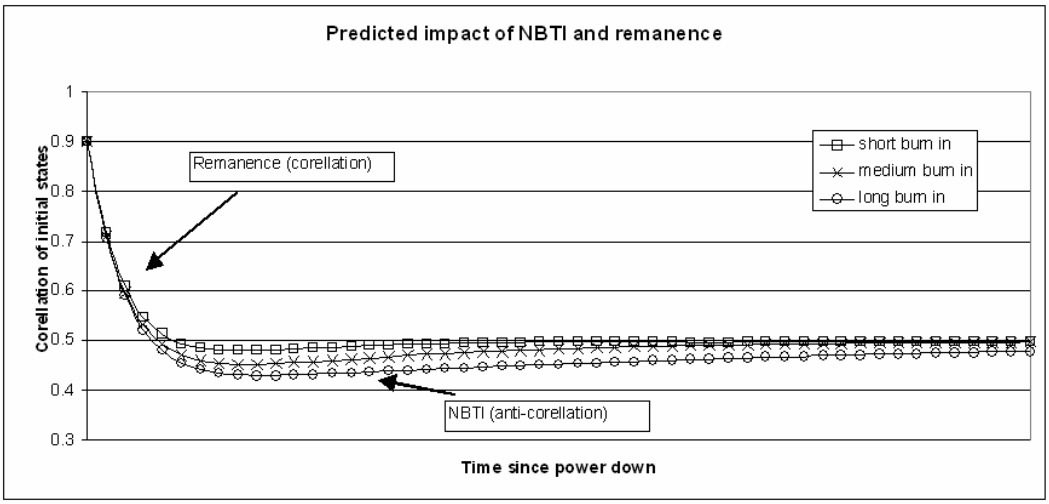

Figure 5.1: Predicted trend in correlation when both data remanence and NBTI are considered.

5.5.3 System Level Design

In FERNS, it is crucial that the same bits used for identification are not also used for randomness; once the bits are transmitted as an identity, their values are known and they have no utility in random number generation. This gives rise to questions about how to best partition the RAM between ID and RNG. One suggestion is given here, based on a work in the field of biometrics in which a mask is used to reduce bitwise error rates to a degree that can be corrected by use of error correcting codes [62]. Consider a bitmask of the same size as the SRAM and stored in non-volatile memory, where each bit of the bitmask corresponds to a bit of SRAM, and contains a 1 if that bit of SRAM is highly reliable, and a 0 otherwise. This bitmask could then be ANDed together with the latent print, producing an ID that is likely imperfect but far more reliable than the initial latent print. If an error correcting code were applied to this more reliable print, the reduced number of errors could be corrected. Additionally, it would ensure that the random bits are excluded from the ID preventing any entropy from being given away.

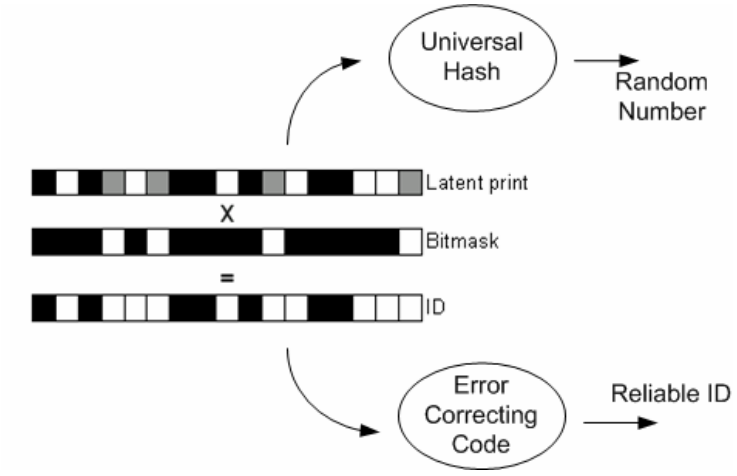

Figure 5.2: A proposed method for partitioning ID and RNG bits from SRAM in FERNS

5.6 Delivered Results

Through experiments and analysis using virtual tags, MSP430 microcontrollers and WISP passive UHF RFID devices, FERNS is shown to be an interesting and viable candidate for accomplishing ID and TRNG in integrated circuits. A paper on FERNS is appearing in the proceedings of the 2007 Conference on RFID security [63]. At this conference, Dan presented the work on July 11, 2007 in Málaga, Spain to an audience of approximately 30 attendees. A patent application for FERNS has been submitted to the UMass

Commercial Ventures and Intellectual Property office. There are plans to submit a comprehensive version of this work to a journal.

APPENDIX

DEBUGGER INDUCED CORELLATION IN MSP430 ID

When observing the MSP430 SRAM state through use of the JTAG debugger, an unexplained correlation between latent prints can be seen. While not offering a potential cause, this appendix offers experimental evidence of this phenomenon. The observed strange behavior was that some latent prints across all chips tended towards a particular pattern. This pattern consists of the memory being split into 4 quadrants, with the first and third quadrants holding 0 and the second and fourth quadrants initializing to 1. According to the addressing scheme of the MSP430, it is addresses 0x0200-0x023F and 0x0280-0x02BF which initialize to 0, while 0x0240-0x027F and 0x02C0-0x02FF initialize to 1.

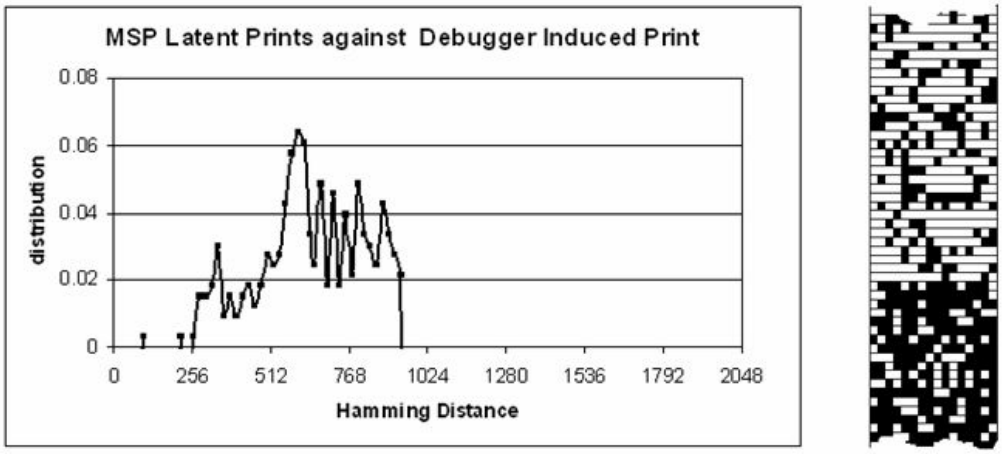

Figure A.1: Latent prints are observed to sporadically favor an identity caused by the JTAG debugger.

Thus, many latent prints are not an accurate representation of the device which generates them; when averaging many latent prints to get a known print, the discovered known print is a combination of the true identity of the device and this common print which all devices occasionally tend towards. Many latent print are thus not very good matches to the known print of the same device.

Two pieces of evidence are offered to support the notion that this is due to the JTAG debugger and not some underlying flaw in analysis. Firstly, the Hamming distance matching between MSP430 known prints and latent prints generated by the same device does not show a Gaussian distribution as would be expected if the discrepancy was caused by noise. An experiment was performed using the WISP hardware,

with the same devices being powered via the debugger and via passive power. When passive power is used, the normal Gaussian shape is observed. When the same addresses on the same locations are powered via the debugger, the outlying points are seen where the latent prints are not reflecting the identity of the device but instead are reflecting the debugger induced identity. This supports the conclusion that the alternate identity is not inherent in the chip, but is instead somehow being caused by the debugger.

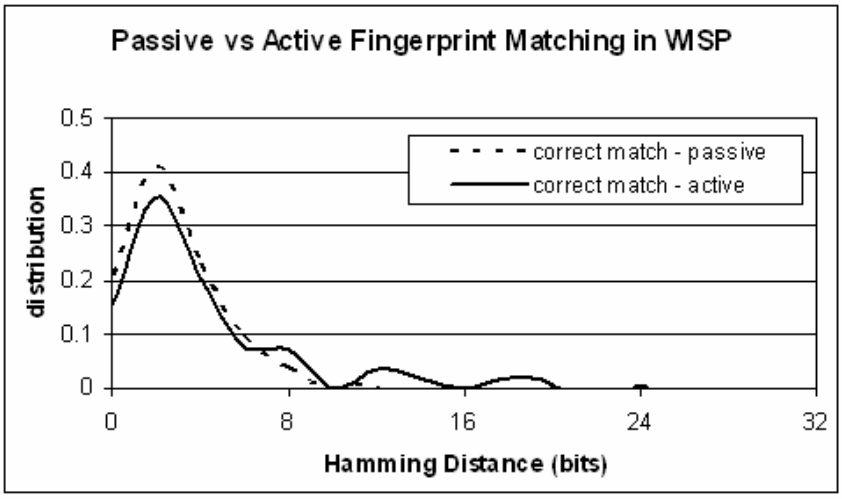

Figure A.2 The same bits of the same device show the debugger induced correlation when actively powered via JTAG, but not when passively powered.

REFERENCES

[1] T. S. Heydt-Benjamin, D. V. Bailey, K. Fu, A. Juels, and T. O'Hare, "Vulnerabilities in first-generation RFID-enabled credit cards." in *Proc. 11th Int. Conf. on Financial Cryptography and Data Security.*, (February 2007).

[2] NXP Semiconductors, "Mifare Standards", http://www.mifare.net/about/standards.asp

[3] International Civil Aviation Organization, "Annex I, Use of Contactless Integrated Circuits In Machine Readable Travel Documents", Version 4.0 May 5, 2002. http://mrtd.icao.int/images/stories/Doc/ePassports/Annex_I_Use_of_Contactless_Integrated_Circuits.pdf

[4] M. C. O'Connor, "Gen 2 EPC Protocol Approved as ISO 18000-6C", RFID Journal, July 11, 2006 http://www.rfidjournal.com/article/articleview/2481/1/1/.

[5] M. C. O'Connor, "Alien Drops Tag Price to 12.9 Cents" RFID Journal, Sept. 15, 2005. http://www.rfidjournal.com/article/articleview/1870/1/1/.

[6] UPM Raflatac white paper, "Tutorial overview of inductively coupled RFID Systems" May 2003. http://www.tayloredge.com/reference/Electronics/rfidsystems.pdf

[7] H. Yan, H. Jianyun, L. Qiang, M. Hao, "Design of Low-power Baseband-processor for RFID Tag" *Int. Symposium on Applications and the Internet Workshops*,. (Phoenix, Arizona, Jan 2006), pp 60-63.

[8] R. Glidden, C. Bockorick, S. Cooper, C. Diorio, D. Dressler, V. Gutnik, C. Hagen, D. Hara, T. Hass, T. Humes, J. Hyde, R. Oliver, O. Onen, A. Pesavento, K. Sundstrom, M. Thomas, "Design of ultra-low-cost UHF RFID tags for supply chain applications," *IEEE Communications Magazine*, vol.42, iss. 8, pp. 140-151, 2004.

[9] A. Ricci, M. Grisanti, I. De Munari, P. Ciapolini, "Design of a Low-Power Digital Core for Passive UHF RFID Transponder". In *Proc. of the 9th EUROMICRO Conference on Digital System Design* (September 2006), pp. 561-568.

[10] H. Soeleman, K. Roy, and B. Paul, "Robust Sub-Threshold Logic for Ultra-Low Power Operation", *IEEE Transactions on VLSI Systems*, Special issue on low-power design, (February 2001), pp. 90-99.

[11] S. Sarma. "Towards the 5 cent Tag," MIT Auto-ID Center white paper, (November 2001)

[12] International Technology Roadmap for Semiconductors, "Design", 2006 Update.

[13] Murfett, D., "The challenge of testing RFID integrated circuits," *2nd IEEE Int. Workshop on Design, Test and Applications*, (January 2004) pp. 410- 412

[14] Ranasinghe, D.C., Lim, D., Cole, P.H., Devadas, S.: A low cost solution to authentication in passive RFID systems. In Cole, P.H., Ranasinghe, D.C., eds.: Networked RFID Systems and Lightweight Cryptography Raising Barriers to Product Counterfeiting. Springer (2007)

[15] S. A. Weis, S. E. Sarma, R. L. Rivest, and D. W. Engels. "Security and Privacy Aspects of Low-Cost Radio Frequency Identification Systems". In Security in Pervasive Computing, (2003), pp. 201-212.

[16] A. Bogdanov, L.R. Knudsen, G. Leander, C. Paar, A. Poschmann, M.J.B. Robshaw, Y. Seurin, and C. Vikkelsoe. "PRESENT: An Ultra-Lightweight Block Cipher", in *Proc. of Cryptographic Hardware and Embedded Systems 2007*, (2007), to appear.

[17] M. Feldhofer, S. Dominikus, and J.Wolkerstorfer. "Strong Authentication for RFID

Systems Using the AES algorithm," *Proc. of Cryptographic Hardware and Embedded Systems 2004*, LNCS, volume 3156, (2004) pp. 357–370

[18] D. Hong, J. Sung, S. Hong, J. Lim, S. Lee, B.-S; Koo, C. Lee, D. Chang, J. Lee, K. Jeong, H. Kim, J. Kim, and S. Chee, "HIGHT: A New Block Cipher Suitable for Low-Resource Device". *In Proc. of Cryptographic Hardware and Embedded Systems 2006*, LNCS, volume 4249, pp 46–59, Springer-Verlag, 2006.

[19] C. Lim and T. Korkishko. "mCrypton – A Lightweight Block Cipher for Security of Low-cost RFID Tags and Sensors". *Workshop on Information Security Applications* , LNCS, volume 3786, (2005) pp. 243-258.

[20] G. Leander, C Paar, A. Poschmann, K Schramm "A Family of Lightweight Block Ciphers Based on DES Suited for RFID Applications". In: *Proceedings of FSE 2007*, to appear.

[21] T. Good, W. Chelton, and M. Benaissa. "Hardware Results for Selected Stream Cipher Candidates". Presented at SASC 2007, (February 2007).

[22] J. Johnson, "Thermal Agitation of Electricity in Conductors", *Phys. Rev.* 32, 97 (1928) – the experiment

[23] H. Nyquist, "Thermal Agitation of Electric Charge in Conductors", *Phys. Rev.* 32, 110 (1928) – the theory

[24] R. Sarpeshkar, T. Delbruck, and C. A. Mead, "White noise in MOS transistors and resistors", IEEE *Circuits Devices Mag*, (November 1993) pp. 23–29.

[25] B. Sunar, W.J. Martin, D. R. Stinson, "A Provably Secure True Random Number Generator with Built-In Tolerance to Active Attacks," *IEEE Transactions on Computers*, (January 2007) pp 109-119

[26] J. E. Neeley, J. G. Harris, "A Subthreshold-CMOS Chaotic Oscillator", in *Proc. of IEEE International Symposium on Circuits and Systems*, (May 1996) pp. 269-271

[27] C. Tokunaga, D. Blaauw, T. Mudge, "A True Random Number Generator with a Metastability-Based Quality Control," *IEEE International Solid-State Circuits Conference*, (February 2007). pp. 404-611

[28] D. J. Kinnimet, E. Chester: "Design of an on-chip random number generator using Metastability". In: *Proceedings of the 28th European Solid-State Circuits Conference*, (2002) pp. 595–598

[29] M. Usami, H. Tanabe, A. Sato, I. Sakama, Y. Maki, T. Iwamatsu, T. Ipposhi, Y. Inoue, "A 0.05x0.05mm RFID chip with easily scaled-down ID-memory". In: *Digest of Technical papers, 2007 IEEE International Solid-State Circuits Conference*, (2007)

[30] P. Friedberg, W. Cheung, C. Spanos, "Spatial variability of critical dimensions." *In: VLSI/ULSI Multilevel Interconnection Conference XXII*. (2005) pp. 539–546

[31] K. Lofstrom, W. Daasch, D. Taylor: "IC identification circuit using device mismatch," In: *Digest of Technical Papers, 2000 IEEE International Solid-State Circuits Conference*. (2000) pp. 372–373

[32] Y. Su, J. Holleman, B. Otis, "A 1.6pJ/bit 96% stable chip ID generating circuit using process variations". In: *Digest of Technical Papers, 2007 IEEE International Solid-State Circuits Conference*. (2007)

[33] B. Gassend, D. Clarke, M. van Dijk, S. Devadas. "Silicon Physical Random Functions," In *Proc. of the 9th ACM conf. on Computer and Communication Security*, (November 2002), pp. 148-160

[34] G.E. Suh, C.W. O'Donnell, I. Sachdev; S. Devadas, "Design and implementation of the AEGIS single-chip secure processor using physical random functions," in: *Proceedings of the 32nd Int. Symposium on Computer Architectur* , (June 2005), pp. 25- 36

[35] X. Tang, V. K. De, J. D. Meindl, "Intrinsic MOSFET Parameter Fluctuations Due to Random Dopant Placement", *IEEE Transactions on VLSI*, Vol. 5, No. 4, (December 1997) pp 369

[36] D. Burnett, K. Erington, C. Subramanian, K. Baker, "Implications of Fundamental Threshold Voltage Variations for High-Density SRAM and Logic Circuits" *Symposium on VLSI Technology*, (June 1994), pp. 15-16

[37] M. Pelgrom, A. Duinmaijer, and A. Welbers, "Matching Properties of MOS Transistors," *IEEE J. Solid-State Circuits*, vol. 24, no. 10, pp. 1433-1440, Oct., 1989.

[38] Predictive Technology Model, "180nm BSIM3 model card for bulk CMOS", (May 2001), available at: http://www.eas.asu.edu/~ptm/

[39] Integrated Silicon Solution, Inc, Datasheet, "IS61LV25616AL - 256K x 16 HIGH SPEED ASYNCHRONOUS CMOS STATIC RAM WITH 3.3V SUPPLY", Rev. E, (February 2006), available at http://www.issi.com/pdf/61LV25616AL.pdf

[40]Altera, "DE2 Development and Education Board, User Manual," version 1.4, (2006), available at: http://www.altera.com/education/univ/materials/boards/DE2_UserManual.pdf

[41] Texas Instruments "SLAU049F - MSP430x1xx Family User's Guide," (2006) available at: http://focus.ti.com/lit/ug/slau049f/slau049f.pdf

[42] Texas Instruments, "SLAU138G -MSP-FET430 Flash Emulation Tool (FET) User's Guide", (May 2007), available at: http://focus.ti.com/lit/ug/slau138g/slau138g.pdf

[43] A.P Sample, D. J.Yeager, P.S. Powledge, J.R. Smith, "Design of a passively powered, programmable platform for UHF RFID systems". *In Proc.s of IEEE Int. Conf. on RFID, 2007*. (2007) pp. 149–156

[44] Smith, J.R., Sample, A., Powledge, P., Roy, S., Mamishev, A.: "A wirelessly-powered platform for sensing and computation". In: *Proceedings of 8th Int. Conf. on Ubiquitous Computing*. (2006) pp. 495–506

[45] D. J. C. MacKay. *Information Theory, Inference, and Learning Algorithms* Cambridge: Cambridge University Press, 2003. ISBN 0-521-64298-1

[46] V. Shoup, *A computational introduction to number theory and algebra, version 1,* Cambridge University Press, 2005. ISBN 0-521-85154-8

[47]B. Barak, R. Shaltiel, E. Tromer, "True random number generators secure in a changing environment," *Proc. of Cryptographic Hardware and Embedded Systems*, (2003), pp. 166-180

[48] N. Nisan, A. Ta-Shma. "Extracting randomness: A survey and new constructions," *Journal of Computer and System Sciences*, vol 58, number 1, (1999). pp. 148-173

[49] Y. Dodis, R. Gennaro, J. Hastad, H. Krawczyk, T. Rabin. "Randomness Extraction and Key Derivation Using the CBC, Cascade and HMAC Modes". In *Proc. of the 24th Annual International Cryptology Conference*, (August 2004), pp. 494–510

[50] Yuksel, K., Kaps, J.P., Sunar, B.: Universal hash functions for emerging ultra-lowpower networks. In: *Proc. of the Communications Networks and Distributed Systems Modeling and Simulation Conference*. (January 2004)

[51] Rukhin et al., “A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications”, *NIST Special Publication 800-22* (revised May 15 2002).

[52] Tanner Consulting & Engineering Services, “Digital Low Power Standard Cell Library for MOSIS TSMC CMOS 0.25 Process”. Available at: http://www.tanner.com/CES/products/design_kits/SampleLibDocs.pdf

[53] J. Guajardo, S. S. Kumar, G. Schrijen, P. Tuyls, “FPGA Intrinsic PUFs and Their Use for IP Protection“, in *Proc. of Cryptographic Hardware and Embedded Systems*, (September 2007), pp. 63-80

[54] M. Denais, V. Huard, C. Parthasarathy, G. Ribes, F. Perrier, N.Revila, A.Bravaix “New methodologies of NBTI characterization eliminating recovery effects*” In Proc. of the 34th European Solid-State Device Research conf.*, (September 2004), pp. 265-268

[55] S. Rangan, N. Mielke and E. Yeh, "Universal Recovery Behavior of Negative Bias Temperature Instability," *IEEE Intl. Electron Devices Mtg.,* (December 2003), p. 341.

[56] C. L. Chen, M. J. Chen, C. J. Wang, K. Wu, ”A New NBTI Lifetime Model (Ig-model) and an Investigation on Oxide Thickness Effect on NBTI Degradation and Recovery” In *Proceedings of the 44th Int. Reliability Physics Symposium*, (March 2006), pp. 741-742

[57] S. V. Kumar, C. H. Kim, and S. S. Sapatnekar, “An Analytical Model for Negative Bias Temperature Instability” *Proceedings of the 2006 IEEE/ACM international conference on Computer-aided design,* (November 2006), pp.493 - 496

[58] W. Wang, S. Yang, S. Bhardwaj, R. Vattikonda, S. Vrudhula, F. Liu, Y. Cao, “The Impact of NBTI on the Performance of Combinational and Sequential Circuits,” *Proceedings of the 44th Design Automation Conference,* (June 2007), pp. 364-369

[59] M. G. Kuhn, R. J. Anderson. ”Low cost attacks on tamper resistant Devices,” *Proceedings of the 5th International Workshop on Security Protocols*, (April 1997), pp. 125–136

[60] P Gutman, “Secure Deletion of Data from Magnetic and Solid-State Memory", *in Proc. of 6th USENIX Security Symposium* (July 1996) pp 77-95

[61] S. Skorobogatov. “Low Temperature Data Remanence in Static RAM”. “Technical Report UCAM-CL-TR-536”, University of Cambridge, Computer Laboratory, (June 2002).

[62] F. Hao, R. Anderson, J. Daugman “Combining Crypto with Biometrics Effectively”, in *IEEE Tran. On Computers*, Vol. 55, No. 9 (September 2006) pp. 1081-1088

[63] D. Holcomb, W.P. Burleson, K. Fu, “Initial SRAM State as a Fingerprint and Source of True Random Numbers for RFID Tags,” in *Proceedings of the Conference on RFID Security*, (July 2007)