Worcester Polytechnic Institute
Digital WPI

Masters Theses (All Theses, All Years)

Electronic Theses and Dissertations

2004-04-28

Universal Hashing for Ultra-Low-Power Cryptographic Hardware Applications

Kaan Yuksel
Worcester Polytechnic Institute

Universal Hashing for Ultra-Low-Power Cryptographic Hardware Applications

by

Kaan Yüksel

A Thesis
Submitted to the Faculty of the
Worcester Polytechnic Institute

in partial fulfillment of the requirements for the
Degree of Master of Science
in

Electrical Engineering

by

April, 2004

Approved:

Dr. Berk Sunar Thesis Advisor ECE Department	Dr. William J. Martin Thesis Committee Mathematical Sciences Department
Dr. Brian King Thesis Committee ECE Department	Prof. Fred J. Looft Department Head ECE Department

Abstract

Message Authentication Codes (MACs) are valuable tools for ensuring the integrity of messages. MACs may be built around a keyed hash function. Our main motivation was to prove that universal hash functions can be employed as underlying primitives of MACs in order to provide provable security in ultra-low-power applications such as the next generation self-powered sensor networks. The idea of using a universal hash function (NH) was explored in the construction of UMAC. This work presents three variations on NH, namely PH, PR and WH. The first hash function we propose, PH, produces a hash of length $2w$ and is shown to be 2^{-w}-almost universal. The other two hash functions, i.e. PR and WH, reach optimality and are proven to be universal hash functions with half the hash length of w. In addition, these schemes are simple enough to allow for efficient constructions. To the best of our knowledge the proposed hash functions are the first ones specifically designed for low-power hardware implementations. We achieve drastic power savings of up to 59% and speedup of up to 7.4 times over NH. Note that the speed improvement and the power reduction are accomplished simultaneously. Moreover, we show how the technique of multi-hashing and the Toeplitz approach can be combined to reduce the power and energy consumption even further while maintaining the same security level with a very slight increase in the amount of key material. At low frequencies the power and energy reductions are achieved simultaneously while keeping the hashing time constant. We develope formulae for estimation of leakage and dynamic power consumptions as well as energy consumption based on the frequency and the Toeplitz parameter t. We introduce a powerful method for scaling WH according to specific energy and power consumption requirements. This enables us to optimize the hash function implementation for use in ultra-low-power applications such as "Smart Dust" motes, RFIDs, and Piconet nodes. Our simulation results indicate that the implementation of WH-16 consumes only 2.95 μW at 500 kHz. It can therefore be integrated into a self-powered device. By virtue of their security and implementation features mentioned above, we believe that the proposed universal hash functions fill an important gap in cryptographic hardware applications.

Preface

Foremost, I would like to thank Prof. Berk Sunar for providing me with the opportunity to be in this position right now. I am more than grateful to him for his continuous support since the very beginning, in terms of technical and personal advice, guidance, inspiration and funding. I feel really lucky that I am going to be able to work with such a great person in the future, too.

In addition, I want to thank my colleague Jens-Peter Kaps for his invaluable help in writing Chapters 5 and 6.

Also, I am glad to have had Prof. William J. Martin and Prof. Brian King in my thesis committee. I appreciate their advice and feedback in spite of tight schedules.

This M.S. Thesis is based upon work supported by the National Science Foundation under Grant No. ANI-0133297.

Kaan Yüksel

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Computing technology is reaching every corner of our lives. Recent advances have enabled developers to embed technology in even tiny devices used in our daily lives, which leads to ubiquitous computing. The trend to ubiquitous computing goes further from mobile communication and personal computing level to ultra-low-power autonomous devices. Here are a few examples of this trend. Piconet [BCE+97] is a general-purpose, low-power ad hoc radio network. It can connect a full range of portable and embedded sensing and computing objects. RFIDs are being used to replace bar codes on goods and to track inventory [SBA00]. "Smart Dust" motes [KRP99] are tiny autonomous nodes containing sensors, transceivers and a power source, and they have limited computing power. Common to all these devices is that they communicate wirelessly and their energy source is extremely limited. Batteries for these devices are tiny and can supply 10 μW for only one day [KRP99]. Moreover, some of these technologies collect energy from environmental sources, such as light, heat, noise, or vibration. Devices that harvest power from environmental sources are commonly referred to as *power scavengers*, and autonomous nodes that use scavengers are called *self-powered*. An implementation of a signal processing unit powered by a large scavenger device that can generate up to 400 μW is described in [AC98]. Newer scavengers are based on microelectromechanical systems (MEMS). They can be integrated on the chip and therefore reduce the cost and size. The scavenger shown in [MMMA+01] produces around 8 μW of energy relying solely on ambient vibration. A major application of this technology is distributed sensor networks. The security aspects of these networks have been reviewed by *NAI Labs* in [CKM00]. However, this study focused only on software implementations on current processors whose energy consumption is far above the amount that can be supplied by a scavenger circuit.

Since power consumption is the vital issue in many applications of these technologies, highly complex cryptographic schemes such as public key cryptography are seldom feasible. Many times, however, what is of primary concern in such systems is protecting the integrity of data. For example, smart dust motes that are embedded in a bridge can monitor stress and inform the authorities in case of emergency. Wireless sensors may monitor plant growth, moisture and PH-value on a farm. In both cases the data is not confidential but its authenticity and integrity are crucial.

For this purpose, digital signature schemes have been proposed [DH76]. However, on low-end computing platforms where processing speed and communication bandwidth are critical, digital signatures may not be the best available choice. Instead, efficient *Message Authentication Codes (MACs)* [Sim92] may be preferable due to their high encryption throughput and short authentication tags.

1.1 Message Authentication Codes

Message Authentication Codes (MACs), also called *keyed hashes*, are used to verify the authenticity of a message, i.e. whether a message did really originate from a given source (with very high probability). Let us suppose *Alice* (the sender of a message) and *Bob* (the recipient) share a secret key. Alice uses the message and the key to compute the MAC, and sends it along with the message. When Bob receives the message, he computes the corresponding MAC, and compares it to Alice's. If they match then he knows that the message is, indeed, from Alice and that nobody has altered it since she sent it, in other words the authenticity of the message is verified.

MACs have the following properties:

1. It is difficult to create a valid MAC for any message without knowing the key.

2. Given a message and the corresponding MAC, it is difficult to create a new message with the same MAC, or any other desired MAC.

3. Given any MAC, it is difficult to find a message that corresponds to that particular MAC.

There are four types of MACs: (1) unconditionally secure [Sti95], (2) stream cipher-based [LRW92], (3) block cipher-based [BKR94] and (4) hash function-based (hash functions introduced in Chapter 3 will underlie this particular type of MACs).

A big disadvantage for both traditional MACs and their counterpart digital signature schemes is that they provide only computational security. This leads to the following consequences:

1. An attacker with sufficient computational power may break the scheme.

2. The lack of a formal security proof makes these schemes vulnerable to possible shortcut attacks.

3. One may need to include a safety margin in his/her design since the exact security level of the scheme may not be known for sure.

Due to their provable security feature in addition to their suitable nature for efficient hardware implementations, we have chosen to focus on *universal hash function*-based MACs rather than their computationally secure counterparts such as MD5 [Riv92] and SHA [Nat02]. We will present mathematical details on provable security in Section 2.2

1.2 Universal Hash Functions

Universal hash functions, first introduced by Carter and Wegman [CW78], provide a solution to the security problems mentioned at the end of the previous section. Roughly speaking, universal hash functions are collections of hash functions that map messages into short output strings such that the collision probability of any given pair of messages is small. A universal hash function family can be used to build an unconditionally secure MAC. For this, the communicating parties share a secret and randomly chosen hash function from the universal hash function family, and a secret encryption key. A message is authenticated by hashing it with the shared secret hash function and then encrypting the resulting hash using the key. Carter and Wegman [CW81] showed that when the hash function family is strongly universal, i.e. a stronger version of universal hash functions where messages are mapped into their images in a pairwise independent manner, and the encryption is realized by a one-time pad, the adversary cannot forge the message with probability better than that obtained by choosing a random string for the MAC. The one-time pad encryption and the hash function selection (from the hash function family) require many key bits, which may be too demanding for most applications. In [Bra83] Brassard observed that combining a universal hash function with a pseudo-random string generator provides a computationally secure message authentication tag with short keys. In this scheme the security of the MAC is dependent on the security of the encryption with the pseudo-random string and the strength of the pseudo-random key used for selecting the hash function.

To our knowledge not much work has been done on improving the performance of universal hashing in hardware. Ramakrishna published a study on the performance of hashing functions in hardware based on universal hashing [RFB94]. However, the main emphasis was on using hash functions for table organization and address translation. In an early work Krawczyk [Kra94] proposed efficient hash functions from a hardware point of view. Considering that a linear feedback shift register (LFSR) can be implemented quite efficiently in hardware, Krawczyk's work introduced two constructions: a CRC-based cryptographic hash function, and a construction based on Toeplitz matrix multiplication. The reference gives a sketch for hardware implementation, which includes a key spreader. However, it is difficult to estimate the power consumption of this function from a sketch. There have been no implementations reported so far. In the past decade we have seen many new hash constructions being proposed, constantly improving in speed and collision probability [Sho96, HK97, Rog95b, Kra95, BHK^{+}99, PR99]. For a survey see [NP99]. However, most of these constructions have targeted efficiency in software implementations, with particular emphasis on matching the instruction set architecture of a particular processor or taking advantage of special instructions made available for multimedia data processing (e.g. Intel's MMX technology). While such high end platforms are essential for everyday computing and communications, in numerous embedded applications space and power limitations prohibit their employment. For instance, smart dust sensor nodes employ a 4-bit or 8-bit low end microprocessor, run the operating system TinyOS [LC02] and are battery powered. These microprocessors do not provide efficient multiplication or variable rotate/shift instructions [PST+02] which are used by many cryptographic functions. Moreover, self-powered sensor nodes commonly employed in RFID tags [SBA00] do not contain a microprocessor but rather a simple control logic. Therefore additional ultra-low-power hardware specifically tailored to perform cryptographic functions might be useful.

Therefore additional ultra-low-power hardware specifically tailored to perform these functions might be useful. Self-powered sensor nodes which do not contain a microprocessor but rather a simple control logic are also common in RFID tags [SBA00].

In [BHK+99] a new hash function family NH for the UMAC message authentication code was introduced. Although NH was intended for efficient and fast implementation in software, we realized that it seems promising for implementation in hardware as well. Therefore, we implemented NH in hardware with an emphasis on low-power and noticed that its power consumption exceeds our limits by far. Instead of optimizing the implementation even more and reducing its power consumption by a fraction we took a different approach. We identified the main power consumers (i.e. registers, adders) and carefully removed those components one by one. We formulated the resulting new algorithms (WH) mathematically and proved that they are still at least as secure as NH. While WH, the variant with the best performance, is consuming an order of magnitude less power than NH, its leakage power consumption still remains a bottleneck. The leakage power is proportional to the circuit size which is proportional to the size of the hash value which, in turn, is proportional to the security level. The technique of multi-hashing was introduced [Rog95a] to increase the security level of a given hash function without changing the size of the hash value at the expense of more key material. We reverse this procedure to preserve the security level while reducing the size of the hash value and therefore the leakage power. We use the Toeplitz approach to reduce the amount of key material needed. The resulting design is scalable and can be tailored to specific energy and power consumption requirements without sacrificing security. We present our implementations of these functions and NH, and compare the results.

1.3 Security Aspects

The security setting of UMAC, described in [BHK$^+$99], is as follows. The parties share two things: a secret and randomly chosen hash function from the universal hash function family, and a secret encryption key. A message is authenticated by hashing it with the shared hash function and then encrypting the resulting hash using the encryption key. Wegman and Carter showed that when the hash function family is strongly universal and the encryption is realized by a one-time pad, the adversary cannot forge with probability better than that obtained by choosing a random string for the MAC. Since the combinatorial property of the universal hash function family is mathematically proven (making no cryptographic hardness assumptions), it needs no "over-design" or "safety margin" the way a cryptographic primitive would. Another benefit of employing a universal hash function comes from the fact that the cryptographic primitive is applied only to the (much shorter) hashed image of the message and therefore, we can select a cryptographically conservative design for this step at the expense of only a minor impact on speed. In the same paper the security of UMAC is rigorously proven, in the sense of giving exact and quantitatively strong results which demonstrate an inability to forge UMAC-authenticated messages assuming an inability to break the cryptographic primitive (pseudo-random function).

In our work we focus on the hash function underlying UMAC not the whole scheme. Hence, we use the term "security" in order to refer to collision probability rather than the exact security of a complete system.

1.4 Thesis Outline

In Chapter 2 the notations used throughout this text will be introduced followed by some concepts about universal hashing as well as the definition of NH, the almost-universal hash function that inspired our investigation.

Chapter 3 starts with the definitions of the three new hash functions and explains how they were developed with an emphasis on low-power hardware implementation. This is followed by the theorems and their proofs showing that the new constructions are at least as secure as NH. Chapter 4 describes how we can further reduce power consumption while maintaining the same level of security by employing the technique of multi-hashing in combination with the Toeplitz approach.

Following that, the detailed implementations of NH and its variants together with illustrative figures are given in Chapter 5. The next chapter presents and discusses the results. Furthermore, we explain how the Toeplitz parameter t can be used to optimize WH with respect to specific energy and power consumption requirements. I would like to mention here that the work presented in these two chapters (Chapters 5 and 6) was made possible by the invaluable help of my colleague Jens-Peter Kaps.

Finally, in Chapter 7 a summary of the work that has been done is given with possible future research on this topic.

Chapter 2

Mathematical Background

The purpose of this chapter is to provide the reader with the notation that will be used throughout the following chapters as well as to introduce the mathematics of universal hashing.

2.1 Notations

Let $\{0,1\}^*$ represent all binary strings, including the empty string. A set $H = \{h : A \rightarrow B\}$, together with some probability distribution, is a family of hash functions with domain $A \subseteq \{0,1\}^*$ of size a and range $B \subseteq \{0,1\}^*$ of size b. The set $C \subseteq \{0,1\}^*$ denotes the finite set of key strings. H_K denotes a single hash function chosen from the set of hash functions H according to a random key $K \in C$. In the text we will set $h = H_K$ to denote a hash function h selected randomly from the set H.

The element $M \in A$ stands for a message string to be hashed and is partitioned into blocks as $M = (m_1, \cdots, m_n)$, where n is the number of message blocks of length w. Similarly the key $K \in C$ is partitioned as $K = (k_1, \cdots, k_n)$, where each block k_i has length w. We use the notation $H[n, w]$ to refer to a hash function family where n is the number of message (or key) blocks and w is the number of bits per block.

Let U_w represent the set of nonnegative integers less than 2^w, and P_w represent the set of polynomials over $GF(2)$ of degree less than w. We will view each message block m_i and key block k_i as belonging to either U_w, P_w or $GF(2^w)$. Here $GF(2^w)$ denotes the finite field of 2^w elements defined by $GF(2)[x]/p(x)$, where $p(x)$ is an irreducible polynomial of degree w over $GF(2)$. Note that in this setting, the bits of a message or key block are associated with the coefficients of a polynomial. Finally, the addition symbol '+' is used to denote both integer and polynomial addition (in a ring or finite field). The meaning should be clear from the context.

2.2 Universal Hashing

A universal hash function, as proposed by Carter and Wegman [CW78], is a mapping from the finite set A with size a to the finite set B with size b. For a given hash function $h \in H$ and for a message pair (M, M') where $M \neq M'$ the following function is defined: $\delta_h(M, M') = 1$ if $h(M) = h(M')$, and 0 otherwise, that is, the function δ yields 1 when the input message pairs collide. For a given finite set H of hash functions $\delta_H(M, M')$ is defined as $\sum_{h \in H} \delta_h(M, M')$, which tells us that $\delta_h(M, M')$ yields the number of functions in H for which M and M' collide. When h is randomly chosen from H and two distinct messages M and M' are given as input, the collision probability is equal to $\delta_h(M, M')/|H|$. We give the definitions of the two classes of universal hash functions used in this paper from [NP99]:

Definition 1 *The set of hash functions $H = \{h : A \rightarrow B\}$ is said to be* ***universal*** *if for every $M, M' \in A$ where $M \neq M'$,*

$$|\{h \in H : h(M) = h(M')\}| \; = \delta_H(M, M') = \frac{|H|}{b} \; .$$

Definition 2 *The set of hash functions $H = h : A \rightarrow B$ is said to be* ***ϵ-almost universal*** *($\epsilon - AU$) if for every $M, M' \in A$ where $M \neq M'$,*

$$|h \in H : h(M) = h(M')| \; = \delta_H(M, M') \leq \epsilon |H| \; .$$

In this definition ϵ is the upper bound for the probability of collision. Observe that the previous definition might actually be considered as a special case of the latter with ϵ being equal to $1/b$. The smallest possible value for ϵ is $(a - b)/(b(a - 1))$.

In the past many universal and almost universal hash families were proposed [Sho96, HK97, Rog95b, Kra95, BHK+99, PR99]. Black et al. introduced an almost universal hash function family called NH in [BHK+99]. The definition of NH is given below.

Definition 3 *([BHK+99]) Given $M = (m_1, \cdots, m_n)$ and $K = (k_1, \cdots, k_n)$, where m_i and $k_i \in U_w$, and for any even $n \geq 2$, NH is computed as follows:*

$$\mathsf{NH}_K(M) = \left[\sum_{i=1}^{n/2} ((m_{2i-1} + k_{2i-1}) \bmod 2^w) \cdot ((m_{2i} + k_{2i}) \bmod 2^w)\right] \bmod 2^{2w} \ .$$

Theorem 1 *([BHK+99]) For any even $n \geq 2$ and $w \geq 1$, $\mathsf{NH}[n, w]$ is 2^{-w}-almost universal on n equal-length strings.*

We refer the reader to the same paper for the proof of the above theorem.

Chapter 3

Variations on NH

In this chapter we introduce three variations to the NH construction. Each one improves upon the previous one in terms of efficiency, and diverges further from NH.

3.1 Definitions

NH - Polynomial (PH) In this construction NH is redefined with message and key blocks as polynomials over $GF(2)$ instead of integers:

Definition 4 *Given $M = (m_1, \cdots, m_n)$ and $K = (k_1, \cdots, k_n)$, where m_i and $k_i \in P_w$, for any even $n \geq 2$, PH is defined as follows:*

$$\mathsf{PH}_K(M) = \sum_{i=1}^{n/2} (m_{2i-1} + k_{2i-1})(m_{2i} + k_{2i}) \ .$$

In a hardware implementation this completely eliminates the carry chain and thereby improves all three efficiency metrics (i.e. speed, space, power) simultaneously. That is, due to the elimination of carry propagations, the operable clock frequency (and thus the speed of the hash algorithm) is dramatically increased. Likewise, the area efficiency is improved since the carry network is eliminated. Finally, due to the reduced switching activity, the power consumption is reduced.

NH-Polynomial with Reduction (PR) The main motivation that led to this construction was to reduce the size of the authentication tag. This is a concern for two reasons. The tag needs to be transmitted along with the data therefore the shorter the tag, the less energy will be consumed for its transmission. The energy consumed by transmitting a single bit can be as high as the energy needed to perform the entire hash computation on the node. The energy needed for transmitting the tag is proportional to its bit-length. Secondly, the size of the tag determines the number of flip-flops needed for storing the tag. The original NH as well as PH introduced above require a large number of flip-flops for the double length hash output. In this construction, the storage and transmission requirement is improved by introducing a reduction polynomial of degree matching the block size, hence reducing the size of the authentication tag by half.

Definition 5 *Given* $M = (m_1, \cdots, m_n)$ *and* $K = (k_1, \cdots, k_n)$*, where* m_i *and* $k_i \in GF(2^w)$*, for any even* $n \geq 2$*, and a polynomial* p *of degree* w *irreducible over* $GF(2)$*,* *PR is defined as follows:*

$$\mathsf{PR}_K(M) = \sum_{i=1}^{n/2}(m_{2i-1} + k_{2i-1})(m_{2i} + k_{2i}) \pmod{p} \ .$$

Note that the original NH construction eliminates the modular reductions used in the previously proposed hash constructions (e.g. MMH proposed in [HK97], SQUARE proposed in [PR99]) since reductions are relatively costly to implement in software. In hardware, however, reductions (especially those with fixed low-weight polynomials) can be implemented quite efficiently.

Weighted NH-Polynomial with Reduction (WH) While processing multiple blocks, it is often necessary to hold the hash value accumulated during the previous iterations in a temporary register. This increases the storage requirement and translates into a larger and slower circuit with higher power consumption. As a remedy we introduce a variant of NH where each processed block is scaled with a power of x. This function is derived from the changes we make to PR which are described in Section 5.4.

Definition 6 *Given $M = (m_1, \cdots, m_n)$ and $K = (k_1, \cdots, k_n)$, where m_i and $k_i \in GF(2^w)$, for any even $n \geq 2$, and an irreducible polynomial $p \in GF(2^w)$, WH is defined as follows:*

$$\mathsf{WH}_K(M) = \sum_{i=1}^{n/2} (m_{2i-1} + k_{2i-1})(m_{2i} + k_{2i})\, x^{(\frac{n}{2}-i)w} \pmod{p} \ .$$

Due to the scaling factor $x^{(\frac{n}{2}-i)w}$, perfect serialization is achieved in the implementation where the new block product is accumulated in the same register holding the hash of the previously processed blocks. This eliminates the need for an extra temporary register as well as other control components required to implement the data path.

3.2 Security Analysis

In this section we give three theorems and their proofs establishing the security of the NH variants.

Theorem 2 *For any even $n \geq 2$ and $w \geq 1$, $\mathsf{PH}[n, w]$ is 2^{-w}-almost universal on n equal-length strings.*

Proof Let M, M' be distinct members of the domain A with equal sizes. We are required to show that

$$Pr\left[\mathsf{PH}_K(M) = \mathsf{PH}_K(M')\right] \leq 2^{-w} \ .$$

Expanding the terms inside the probability expression, we must prove

$$Pr\left[\sum_{i=1}^{n/2}(m_{2i-1}+k_{2i-1})(m_{2i}+k_{2i}) = \sum_{i=1}^{n/2}(m'_{2i-1}+k_{2i-1})(m'_{2i}+k_{2i})\right] \leq 2^{-w} \ . \quad (3.1)$$

The probability is taken over uniform choices of $(k_1, k_2, \ldots, k_n)$ with each $k_i \in P_w$ and the arithmetic is over $GF(2)$. Since M and M' are distinct, $m_i \neq m'_i$ for some

$1 \leq i \leq n$. Addition and multiplication in this ring, i.e. $GF(2)[x]$, are commutative, hence there is no loss of generality in assuming $m_2 \neq m'_2$. Now let us prove that for any choice of $k_2, k_3, \ldots, k_n$ we have

$$Pr_{k_1 \in P_w}\left[(m_1+k_1)(m_2+k_2)+\sum_{i=2}^{n/2}(m_{2i-1}+k_{2i-1})(m_{2i}+k_{2i}) = \right.$$

$$\left.(m'_1+k_1)(m'_2+k_2)+\sum_{i=2}^{n/2}(m'_{2i-1}+k_{2i-1})(m'_{2i}+k_{2i})\right] \leq 2^{-w}$$

which will imply (3.1). Let

$$y = \sum_{i=2}^{n/2}(m'_{2i-1}+k_{2i-1})(m'_{2i}+k_{2i}) - \sum_{i=2}^{n/2}(m_{2i-1}+k_{2i-1})(m_{2i}+k_{2i}) \ .$$

Rewriting the probability yields

$$Pr_{k_1}\left[(m_1+k_1)(m_2+k_2)-(m'_1+k_1)(m'_2+k_2)=y\right] \leq 2^{-w} \ .$$

Next, we show that for any m_2, m'_2 and $y \in P_w$ there exists at most one $k_1 \in P_w$ such that

$$k_1(m_2-m'_2)+m_1(m_2+k_2)-m'_1(m'_2+k_2)=y \ .$$

Then the identity becomes

$$k_1(m_2-m'_2) = y - m_1(m_2+k_2)+m'_1(m'_2+k_2) \ . \tag{3.2}$$

Since $m_2 \neq m'_2$, the term $(m_2 - m'_2)$ cannot be zero. The analysis can be concluded by examining two possible cases. Since there is no zero divisor in $GF(2)[x]$, either

$(m_2 - m'_2)$ divides the right hand side of (3.2), i.e., $(m_2 - m'_2)$ is a factor of the right hand side, and there is one $k_1 \in P_w$ satisfying the equation, which is

$$k_1 = \left(y - m_1(m_2 + k_2) + m'_1(m'_2 + k_2)\right) / (m_2 - m'_2) \ ,$$

or $(m_2 - m'_2)$ does not divide the right hand side of (3.2) and there is no $k_1 \in P_w$ satisfying this equation. These two cases prove that there can be at most one k_1 value (out of 2^w possible values), which causes collision. Therefore,

$$Pr\left[\mathsf{PH}_K(M) = \mathsf{PH}_K(M')\right] \leq 2^{-w} \ .$$

□

Theorem 3 *For any even $n \geq 2$ and $w \geq 1$, $\mathsf{PR}[n, w]$ is universal on n equal-length strings.*

Proof Let M, M' be distinct members of the domain A with equal lengths. We are required to show that

$$Pr\left[\mathsf{PR}_K(M) = \mathsf{PR}_K(M')\right] = 2^{-w} \ .$$

Expanding the terms inside the probability expression, we obtain

$$Pr\left[\sum_{i=1}^{n/2}(m_{2i-1} + k_{2i-1})(m_{2i} + k_{2i}) = \sum_{i=1}^{n/2}(m'_{2i-1} + k_{2i-1})(m'_{2i} + k_{2i}) \pmod p\right] = 2^{-w} \ .$$

We proceed as in the proof of Theorem 2 with the only exception of the arithmetic performed in $GF(2^w)$, instead of P_w. Similarly the derivation yields

$$k_1(m_2 - m'_2) = y - m_1(m_2 + k_2) + m'_1(m'_2 + k_2) \pmod p \ .$$

Since $m_2 \neq m_2'$, the term $(m_2 - m_2')$ cannot be zero and its inverse in $GF(2^w)$ exists. Hence there is exactly one $k_1 \in GF(2^w)$ satisfying the equation, which is

$$k_1 = (m_2 - m_2')^{-1} \left(y - m_1(m_2 + k_2) + m_1'(m_2' + k_2)\right) \pmod{p} .$$

Therefore,

$$Pr\left[\mathsf{PR}_K(M) = \mathsf{PR}_K(M')\right] = 2^{-w} .$$

□

Theorem 4 *For any even $n \geq 2$ and $w \geq 1$, $\mathsf{WH}[n, w]$ is universal on n equal-length strings.*

Proof Let M, M' be distinct members of the domain A with equal lengths. We are required to show that

$$Pr\left[\mathsf{WH}_K(M) = \mathsf{WH}_K(M')\right] = 2^{-w} .$$

Expanding the terms inside the probability expression, we obtain

$$Pr\left[\sum_{i=1}^{n/2}(m_{2i-1} + k_{2i-1})(m_{2i} + k_{2i})\left(x^{(\frac{n}{2}-i)w}\right) = \sum_{i=1}^{n/2}(m_{2i-1}' + k_{2i-1})(m_{2i}' + k_{2i})\left(x^{(\frac{n}{2}-i)w}\right) \pmod{p}\right] = 2^{-w} . \quad (3.3)$$

The probability is taken over uniform choices of $(k_1, \ldots, k_n)$ with each $k_i \in GF(2^w)$ and the arithmetic is over $GF(2^w)$. Since M and M' are distinct, $m_i \neq m_i'$ for some

$1 \leq i \leq n$. Let $m_{2l} \neq m'_{2l}$. For any choice of $k_1, \ldots, k_{2l-2}, k_{2l}, \ldots, k_n$ having

$$Pr_{k_{2l-1} \in GF(2^w)} \left[\sum_{i=1}^{n/2} (m_{2i-1} + k_{2i-1})(m_{2i} + k_{2i}) \left(x^{(\frac{n}{2}-i)w} \right) = \right.$$

$$\left. \sum_{i=1}^{n/2} (m'_{2i-1} + k_{2i-1})(m'_{2i} + k_{2i}) \left(x^{(\frac{n}{2}-i)w} \right) \pmod{p} \right] = 2^{-w} \quad (3.4)$$

satisfied for all $1 \leq l \leq n/2$ implies (3.3). Setting y and z as

$$\begin{aligned} y &= \left[\sum_{i=1}^{l-1} (m'_{2i-1} + k_{2i-1})(m'_{2i} + k_{2i}) x^{(\frac{n}{2}-i)w} \right. \\ &\quad \left. - \sum_{i=1}^{l-1} (m_{2i-1} + k_{2i-1})(m_{2i} + k_{2i}) x^{(\frac{n}{2}-i)w} \right] \pmod{p} \end{aligned}$$

and

$$\begin{aligned} z &= \left[\sum_{i=l+1}^{n/2} (m'_{2i-1} + k_{2i-1})(m'_{2i} + k_{2i}) x^{(\frac{n}{2}-i)w} \right. \\ &\quad \left. - \sum_{i=l+1}^{n/2} (m_{2i-1} + k_{2i-1})(m_{2i} + k_{2i}) x^{(\frac{n}{2}-i)w} \right] \pmod{p} \end{aligned}$$

we rewrite the probability bound in (3.4) as

$$Pr_{k_{2l-1}} \left[x^{(\frac{n}{2}-i)w} \left[(m_{2l-1} + k_{2l-1})(m_{2l} + k_{2l}) - (m'_{2l-1} + k_{2l-1})(m'_{2l} + k_{2l}) \right] \right.$$

$$\left. = y + z \pmod{p} \right] = 2^{-w} .$$

Since $x^{(\frac{n}{2}-i)w}$ is invertible in $GF(2^w)$, the equation inside the probability expression can be rewritten as follows.

$$k_{2l-1}(m_{2l} - m'_{2l}) + m_{2l-1}(m_{2l} + k_{2l}) - m'_{2l-1}(m'_{2l} + k_{2l}) = x^{-(\frac{n}{2}-i)w}(y + z) \pmod{p}$$

Solving the equation for k_{2l-1}, we end up with the following

$$k_{2l-1} = (m_{2l} - m'_{2l})^{-1} \left((x^{-(\frac{n}{2}-i)w})(y+z) - m_{2l-1}(m_{2l} + k_{2l}) + m'_{2l-1}(m'_{2l} + k_{2l})\right) \pmod{p} .$$

Note that $(m_{2l} - m'_{2l})$ is invertible since in the beginning of the proof we assumed that $m_{2l} \neq m'_{2l}$. This proves that for any m_{2l}, m'_{2l} (with $m_{2l} \neq m'_{2l}$) and y, $z \in GF(2^w)$ there exists exactly one $k_{2l-1} \in GF(2^w)$ which causes a collision. Therefore,

$$Pr\left[\mathsf{WH}_K(M) = \mathsf{WH}_K(M')\right] = 2^{-w} .$$

□

Chapter 4

Toeplitz Approach

This chapter consists of the details of how we can achieve further improvements in the newly introduced hash functions utilizing the well-known Toeplitz approach. Our main motivation here is to reduce the leakage power.

We know that the power consumed by a VLSI circuit has two components: Leakage power and dynamic power. Only the latter depends on frequency. This means that at lower frequencies the total power consumption is dominated by the leakage power consumption. Since this component is directly proportional to the size of the circuit, we now aim to design a smaller circuit. The circuit size scales with the data path width, i.e. the block size w of the message and the key. Since the collision probability is equal to 2^{-w} (see Section 3.2), reducing the block size w will significantly increase this probability and impair the security of the system. In order to decrease the collision probability without changing the word size, [BHK$^+$99] uses the technique of multi-hashing [Rog95a] in which different random members of the hash function family are applied to the message, and the results are concatenated to form the hash value. We use a similar approach, however, we preserve the collision probability while reducing the word size. For instance, to obtain the collision probability of 2^{-w} with a block size of $w/4$ bits, each message block is hashed 4 times with independent keys. The computed hash outputs ($w/4$ bits each) are then concatenated to form the w bit hash result. The drawback of this method is that it requires 4 times the key material. As a remedy one can employ the well-known Toeplitz approach [MNT90, BHK$^+$99, Kra94] in which shifted versions of one key rather than several independent keys are used. In this case, however, since the keys are related to each other, it is not obvious that the collision probability can be maintained. In Theorem 5 we will prove that the Toeplitz construction for WH can still achieve the desired result.

4.1 Construction

The hash function family $WH^T[n, w, t]$ ("Toeplitz-WH") has three parameters, namely n, w and t. The additional parameter t stands for Toeplitz iteration count, where $t \geq 1$, and the others are defined as before. Domain A remains the same whereas the range is now $B = \{0, 1\}^{wt}$. A function is selected by a key K of length $w(n+2(t-1))$ bits. In other words, K is composed of $(n + 2(t - 1))$ w-bit words. We have

$K = (k_1, k_2 \cdots, k_{n+2(t-1)})$, where each k_i is a w-bit word. The notation $K_{i..j}$ represents $K = (k_i, k_{i+1}, \cdots, k_j)$. Then for a message string $M \in A$, $WH^T_K(M)$ is defined as follows.

$$WH^T_K(M) = (WH_{K_{1..n}}(M), WH_{K_{3..n+2}}(M), \cdots, WH_{K_{2t-1..n+2t-2}}(M)).$$

Theorem 5 *For any $w \geq 1$, $t \geq 1$, and any even $n \geq 2$, $WH^T[n, w, t]$ is universal on equal-length strings with collision probability of 2^{-wt}.*

Proof For the sake of brevity we will use WH and WH^T instead of $WH[n, w]$ and $WH^T[n, w, t]$, respectively. Let M and M' be distinct members of the domain A with equal lengths. We are required to show

$$Pr[WH^T_K(M) = WH^T_K(M')] = 2^{-wt} \tag{4.1}$$

We have $M = (m_1, m_2, \cdots, m_n)$, $M' = (m'_1, m'_2, \cdots, m'_n)$ and $K = (k_1, k_2, \cdots, k_{n+2(t-1)})$, where m_i, m'_i and k_i are all w bit words associated with polynomials. Note that the arithmetic is carried out over $GF(2^w)$ with the irreducible polynomial p of degree w. Next we define the event E_j for $j \in \{1, \cdots, t\}$ as follows.

$$E_j : \sum_{i=1}^{n/2}(k_{2i+2j-3} + m_{2i-1})(k_{2i+2j-2} + m_{2i})x^{(\frac{n}{2}-i)w} =$$
$$\sum_{i=1}^{n/2}(k_{2i+2j-3} + m'_{2i-1})(k_{2i+2j-2} + m'_{2i})x^{(\frac{n}{2}-i)w} \pmod{p}$$

We call each term in the summations of the E_j a "clause" (e.g., $(k_1 + m_1)(k_2 + m_2)x^{(\frac{n}{2}-1)w}$ is a clause). Now the probability in (4.1) can be rewritten as

$$Pr[E_1 \cap E_2 \cap \cdots \cap E_t] \ .$$

Without loss of generality, we can assume that M and M' disagree in the last clause (i.e., $m_{n-1} \neq m'_{n-1}$ or $m_n \neq m'_n$). Notice that if M and M' agreed in the last clause then each E_j would be satisfied if and only if it was also satisfied when that last clause was omitted. Hence, we could truncate M and M' after the last clause in which they disagree, and still obtain exactly the same set of keys causing collisions.

Now, again without loss of generality, we can assume that $m_{n-1} \neq m'_{n-1}$ because for each iteration of E_j the key is shifted by two words making the case symmetric. We proceed by proving that for all $j \in \{1, \cdots, t\}$, $Pr[E_j \ is \ true \ \ | \ E_1, \cdots, E_{j-1} \ are \ true] = 2^{-w}$, implying the theorem.

For $j = 1$, the claim is satisfied due to Theorem 4. For $j > 1$, the events E_1 through E_{j-1} depend only on key words $k_1, \cdots, k_{n+2j-4}$ while E_j depends also on k_{n+2j-3} and k_{n+2j-2}. By fixing k_1 through k_{n+2j-4} such that E_1 through E_{j-1} are satisfied, and fixing any value for k_{n+2j-3}, we prove that there is only one value of k_{n+2j-2} satisfying E_j. Let

$$\begin{aligned} y = \sum_{i=1}^{n/2-1} (k_{2i+2j-3} + m'_{2i-1})(k_{2i+2j-2} + m'_{2i})x^{(\frac{n}{2}-i)w} - \\ \sum_{i=1}^{n/2-1} (k_{2i+2j-3} + m_{2i-1})(k_{2i+2j-2} + m_{2i})x^{(\frac{n}{2}-i)w} \ . \end{aligned}$$

Thus, E_j becomes

$$E_j : (k_{n+2j-3}+m_{n-1})(k_{n+2j-2}+m_n)-(k_{n+2j-3}+m'_{n-1})(k_{n+2j-2}+m'_n) = y \pmod p \ .$$

Now we are required to prove that

$$Pr[(k_{n+2j-3}+m_{n-1})(k_{n+2j-2}+m_n)-(k_{n+2j-3}+m'_{n-1})(k_{n+2j-2}+m'_n) = y \pmod{p}] = 2^{-w} \ .$$

Solving the equation inside the above probability expression for k_{n+2j-2}, we end up with the following

$$k_{n+2j-2} = (m_{n-1}-m'_{n-1})^{-1} \big(y - m_n(m_{n-1}+k_{n+2j-3}) + m'_n(m'_{n-1}+k_{n+2j-3})\big) \pmod p \ .$$

Note that $(m_{n-1}-m'_{n-1})$ is invertible since in the beginning of the proof we assumed $m_{n-1} \neq m'_{n-1}$. This proves that for any k_{n+2j-3}, m_{n-1}, m'_{n-1} (with $m_{n-1} \neq m'_{n-1}$) $\in GF(2^w)$ there exists exactly one $k_{n+2j-2} \in GF(2^w)$ which causes a collision. Therefore,

$$Pr[WH^T_K(M) = WH^T_K(M')] = 2^{-wt} \ .$$

$\square$

Chapter 5

Implementations

This chapter discusses the design process that has been followed in the course of realizing the NH-variants as well as the NH construction itself.

The power dissipation in CMOS devices can be summarized by the following formula [DM95]:

$$P = \underbrace{\left(\frac{1}{2} \cdot C \cdot V_{DD}^2 + Q_{se} \cdot V_{DD}\right) \cdot f \cdot N}_{\mathrm{P_{Dynamic}}} + \underbrace{I_{leak} \cdot V_{DD}}_{\mathrm{P_{Leakage}}} \tag{5.1}$$

The term $\mathrm{P_{Dynamic}}$ represents the power required to charge and discharge circuit nodes as well as the power dissipation during output transitions. The terms C, Q_{se}, and V_{DD} are technology dependent [DM95]. The switching activity, i.e. the number of gate output transitions per clock cycle, is represented by N and the operating frequency by f. The second term $\mathrm{P_{Leakage}}$ represents the static power dissipation due to the leakage current I_{leak}. The leakage current is directly determined by the number of gates and the fabrication technology. For more information about low-power design see [RP96]. In order to minimize the power consumption, we designed our CMOS circuits according to the following rules:

- The number of transitions (‘0’ to ‘1’ and ‘1’ to ‘0’) has to be minimal.
- The circuit size should be minimized.
- Glitches cause unnecessary transitions and therefore should be avoided.

5.1 NH

The algorithm for NH is described in [BHK+99] and also given in Definition 3 as

$$\mathsf{NH}_K(M) = \left[\sum_{i=1}^{n/2} ((m_{2i-1} + k_{2i-1}) \bmod 2^w) \cdot ((m_{2i} + k_{2i}) \bmod 2^w)\right] \bmod 2^{2w} .$$

This leads to the simplified block diagram shown in Figure 5.1. The actual block diagram for the circuit is much more complex and can be found in Figure 5.2. The message and the key are assumed to be split into n blocks of w bits. Messages that are shorter than a multiple of $2 \cdot w$ are padded. All odd message blocks are applied to input $m1$, all even message blocks to input $m2$. The blocks of the key are applied similarly to $k1$ and $k2$. The final adder accumulates all $n/2$ products.

The output of Adder 1 is $ma = m1 + k1 \bmod 2^w$, the output of Adder 2 is $mb = m2 + k2 \bmod 2^w$. These are integer additions where the carry out is discarded. The

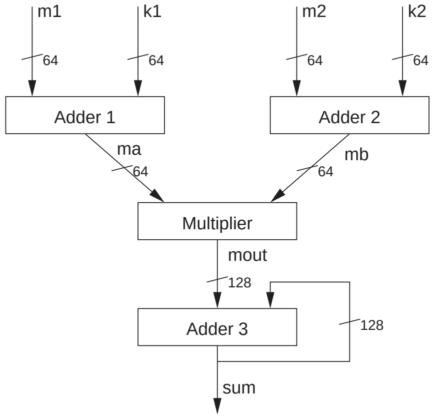

Figure 5.1: Simplified Functional Diagram for NH

multiplication results in $mout = ma \cdot mb$. For each multiplication of two w-bit numbers, w partial products need to be computed and added: $mout = \sum_{j=1}^{w} ma \cdot mb[j] \cdot 2^{j-1}$. As power consumption is our main concern and not speed we chose to implement a bit serial multiplier. It computes one partial product during each clock cycle and adds it to the sum of the previous partial products using the Right Shift Algorithm [Par00].

A bit serial adder produces one bit of the result with each clock cycle, starting with the LSB and it has minimal glitching. We used a bit serial adder for Adder 2 as its result can directly be used by the bit serial multiplier. However, the multiplicand has to be available immediately. Therefore we used a simple ripple carry adder to implement Adder 1. Its main disadvantage is that it takes a long time until the carries propagate through the adder, causing a lot of glitching and therefore a high power consumption. However, Adder 1 needs to compute a new result only every 64 clock cycles, hence its dynamic power consumption is tolerable. The addition of the partial products is accomplished using a carry-save adder. This adder uses the redundant carry-save notation which results in minimal glitching as the carries are not fully propagated. However, 64 additional flip-flops are required to store the carry bits.

After one multiplication has been computed, its result has to be added to the accumulation of the previous multiplications as indicated by Adder 3 in Figure 5.1. Rather than having a separate multiplier and adder, in the actual implementation we add the partial products of the next multiplication immediately to the result of the previous additions. This technique stores the result of Adder 3 in the Multiplier thus saving a 128 bit register and a 128 bit multiplexer.

The carry-save adder has separate data paths for sum and carry. It can add the partial products of one multiplication very efficiently. However, after the product is computed it needs to be re-aligned before the partial products of the next multiplication can be added to this result. This re-alignment involves converting the number from carry-save notation to standard binary notation, i.e., adding the carries to the sum. This addition is done using a ripple carry adder (Figure 5.2 shows that this Ripple Carry Adder has the signal rcasum as output). Even though the products of the multiplication are 128 bits wide, the carry is only 64 bits wide, hence the ripple carry adder is only 64 bits wide. This sum needs to be computed only after a multiplication has finished, i.e., every 64 clock cycles. As the result is not needed during the other 63 clock cycles, we isolate the operands from the ripple carry adder, hence the adder does not consume power due to switching activity when its output is not needed. After one multiplication is completed and the result is re-aligned, the carry registers are set to zero for the next computation.

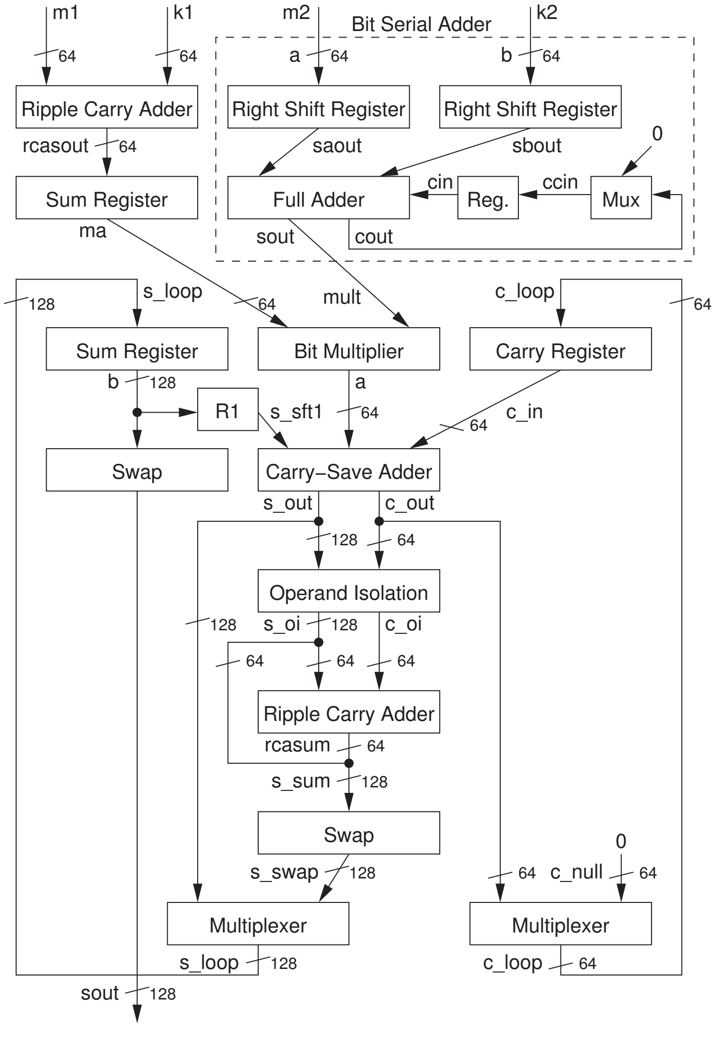

Figure 5.2: Block Diagram for NH

We use this implementation of the original NH algorithm as a reference for comparison with its variations described below.

5.2 NH-Polynomial (PH)

The main power consumers in the implementation of NH are the ripple carry adders and flip-flops needed for the multiplier and the bit serial adder. PH is a variation on NH in that it uses polynomials over $GF(2)$ instead of integers. This replaces the costly adders with simple XOR gates which consume significantly less power. Adder 1 is replaced by 64 XOR gates. The bit-serial adder (Adder 2) is replaced by XORs and a 64 bit shift register which reduces the complexity of this unit by 64 flip-flops. The Multiplier and Adder 3 are combined as in NH but the carry-save adders are replaced by XORs. As there are no carries another 64 flip-flops are saved. Just changing NH from using integers to polynomials reduces the number of cells by 65%, the dynamic power consumption by 38% and the leakage power by more than a half.

5.3 NH-Polynomial with Reduction (PR)

The main difference between PR and PH is that the result is reduced to 64 bits using an irreducible polynomial. In our hardware implementation the multiplication and the reduction are interleaved. This makes the reduction very efficient. Moreover, using low Hamming-weight polynomials the reduction can be achieved with only a few gates and minimal extra delay. However, the Multiplier and Adder 3 can no longer be merged. Therefore, we are not able to reduce the number of flip-flops in our implementation but we reduced the switching activity as Adder 3 computes a new result only once every 64 clock cycles. The number of cells for this implementation is slightly higher than for PH and thus the leakage power is increased. Due to the reduced switching activity, however, the dynamic power consumption is now 50% less than that of NH.

5.4 Weighted NH-Polynomial with Reduction (WH)

This design was inspired by the bottlenecks we observed in the implementation of PR. For instance, the Multiplier and Adder 3 (Figure 5.1) could not be merged as in PH. We removed from PR's implementation a 64 bit register, a 64 bit multiplexer and the XOR gates of Adder 3. The function of the resulting design is characterized by the construction shown in Definition 6. Compared to NH, the removal of the mentioned components reduced the dynamic power consumption by 59%, the leakage power consumption by 66%, and the number of cells by 74%. These dramatic savings become more obvious when the block diagrams for NH in Figure 5.2 and for WH in Figure 5.3 are compared.

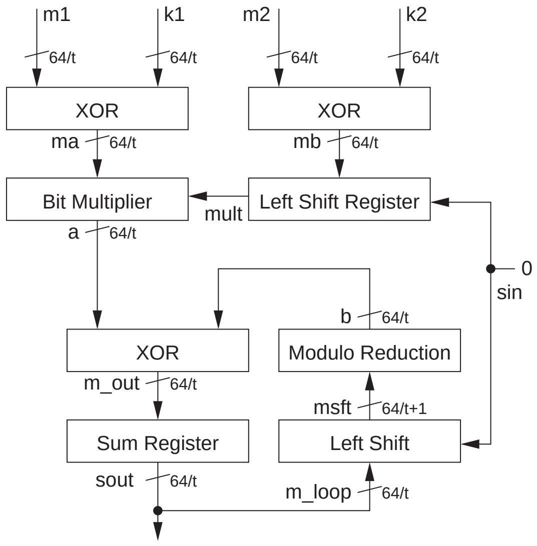

Figure 5.3: Detailed Block Diagram for WH Datapath Depending on Toeplitz Parameter t

5.5 WH with Toeplitz Construction

We have shown in Section 4 that it is possible to preserve the security level while reducing the word size if the message is hashed multiple times with independent keys. The Toeplitz approach describes how these keys can be generated efficiently. For our implementation we assume that the circuit, which generates the messages and the keys, implements this approach and delivers keys and the appropriate parts of the message to our hash function implementation.

Figure 5.3 shows a detailed block diagram for WH depending on the Toeplitz parameter t. We define the word size w as 64 bits. The block size is the word size divided by the Toeplitz parameter t. The implementation of WH with a 64 bit word size, i.e. $t = 1$, is called WH-64. The minimum input length in this case is $2 \cdot w = 128$ bits. Half of these bits are applied to m1 and the other half to m2. The same holds for the key. In order to achieve the same level of security for a word size of 32 bits we would hash the message twice. Hence, the Toeplitz iteration count t would be two. The implementation of this is called WH-32. In order to hash the same input of 128 bits WH-32 would need to compute four hashes. The length of the final output is the same.

5.6 Control Logic

The control logic manages the switching of the multiplexers, loading of the next data set and resetting the carry registers. Due to the iterative nature of the multiplier, the control logic requires a counter. Traditionally, counters are built using a register and a combinational incrementer. The incrementer requires long carry propagations, which cause glitching and introduce latency. As the critical delay of the datapath is minimized in our design to only a few levels of logic, the delay of an incrementer would create a bottleneck in the control circuit. Hence, optimization of this unit is essential. Instead of an integer counter, we use a linear feedback shift register (LFSR) with 6 flip-flops, enhanced to “count” up to 64. LFSRs have minimal glitching and therefore make power efficient and fast counters. The control logic for WH-32 and

WH-16 uses the same principle and “counts” only to 32 and 16.

Chapter 6

Results

This chapter presents and discusses the results obtained from the synthesis tools for a number of different parameter settings. In particular, this means power, delay and area efficiency for several configurations with different choices of hash functions, frequency and the wordsize w (as the Toeplitz iteration count t varies). The *maximum delay* determines the highest operable frequency.

For synthesizing our designs we used the Synopsys tools Design Compiler [Syn02a] and Power Compiler [Syn02b], and the TSMC 0.13 μm ASIC library. The results of the simulation on many input sets were verified with the Maple package [HHR98] for consistency.

6.1 Comparing NH to Its Variants

Table 6.1 lists power, area, and delay results of the hash function implementations, synthesized for operation at 100 MHz. As shown in this table, the best performance is achieved by WH.

Table 6.1: Comparison of Hash Function Implementations at 100 Mhz

Design	**Dynamic Power**		**Leakage Power**		**Number of Cells**		**Total Cell Area**		**Maximum Delay**	
	μW	%	μW	%		%	μm^2	%	**ns**	**speedup**
NH	1093.9	100	28.1	100	1576	100	26943	100	9.92	1.0
PH	682.7	62	12.1	43	557	35	11997	45	1.35	7.4
PR	549.9	50	14.0	50	616	39	12919	48	1.35	7.4
WH	452.3	41	9.4	33	412	26	8662	32	1.35	7.4

When run at 100 MHz, WH consumes 41% of the dynamic power and 33% of the leakage power consumed by NH and at the same time occupies only 32% of the area[1] and WH can run 7.4 times faster than NH. We proved that it can provide the same level of security, though. However, the dynamic power consumption of WH at this frequency is 452.3 μW. This is much higher than our aim of 20 μW. The CMOS power formula in Equation 5.1 shows that the dynamic power consumption is directly proportional to the operating frequency. Hence, the implementations consume $1/200^{th}$ of the dynamic power when clocked at 500 kHz (this lower frequency is used in sensor node implementations [AC98]), however, the leakage power remains the same. For this reason the leakage power, at low speeds, becomes the limiting factor for ultra-

[1] The area is given in terms of two input equivalent NAND gates.

low-power implementations as demonstrated on Table 6.2. This leads us to look for ways to reducing the leakage power by using various block sizes with Toeplitz Approach (see Chapter 4 for mathematical details) as depicted for WH construction in Sections 6.2 and 6.3.

WH can operate with as little as 11.6 μW and this is in the range of the power produced by a MEMS scavenger [MMMA+01]. However, we would like to note that we used an ASIC standard cell library to obtain these results. A full custom IC-design would yield even higher power savings.

Table 6.2: Comparison of Power Consumption at 100 Mhz and 500 kHz

	100 MHz						**500 kHz**					
Design	**Dynamic**		**Leakage**		**Total**		**Dynamic**		**Leakage**		**Total**	
	μW	%	μW	%	μW	%	μW	%	μW	%	μW	%
NH	1093.9	100	28.1	100	1122.0	100	5.47	100	28.1	100	33.6	100
PH	682.7	62	12.1	43	694.8	62	3.41	62	12.1	43	15.5	46
PR	549.9	50	14.0	50	563.9	50	2.75	50	14.0	50	16.8	50
WH	452.3	41	9.4	33	461.7	41	2.26	41	9.4	33	**11.6**	35

6.2 WH with Various Block Sizes

We implemented WH with block size w of 64 bits (WH-64), 32 bits (WH-32), and 16 bits (WH-16). Table 6.3 shows the results for power, area, and delay for these hash implementations, synthesized for operation at 100 MHz .

It can be seen in Table 6.3 that the dynamic and leakage power consumptions as

Table 6.3: Comparison of WH Implementations at 100 Mhz

Design	**Dynamic Power**		**Leakage Power**		**Circuit Area**		**Maximum Delay**	
	μW	%	μW	%	**gates**	%	**ns**	**speedup**
WH-64	452.3	100	9.36	100	1701	100	1.35	1.0
WH-32	217.5	48	4.81	51	873	51	1.31	1.0
WH-16	126.2	28	2.32	25	460	27	0.76	1.8

well as the circuit size are reduced almost linearly with the block size. We analytically verify these observations. For simplicity, in our analysis we ignore the contributions of the control and reduction units to the power consumption. From the power dissipation formula for CMOS (Equation 5.1) we see that the leakage power is proportional to the number of gates (i.e. area A) used: $P_{Leak} \propto A$. The area in turn is proportional to the block size, i.e. $A \propto w$, and therefore

$$P_{Leak} \propto w \ .$$

The dynamic power consumption is proportional to the operating frequency and the number of logic transitions: $P_{Dyn} \propto f\,N$. Since $N \propto w$, the dynamic power consumption scales with the frequency and the block size as follows.

$$P_{Dyn} \propto f\,w$$

The total power consumption $P = P_{Dyn} + P_{Leak}$ is related to f and w as

$$P \propto w(cf + 1) \ .$$

Here c is a fixed constant factor. The energy E consumed is the total power times the running time: $E = PT$. Since $T = \frac{w}{f}$, the total energy consumption is related to the block size and the frequency as

$$E \propto w^2 \left(c + \frac{1}{f} \right) \ .$$

The slight nonlinearity observed in Table 6.3 can be explained by considering the control and the modulo reduction units, which are the only parts in the circuit that do not scale linearly with the block size. The size of the modulo reduction unit depends on the primitive polynomial and can be made negligible by utilizing a low-Hamming weight polynomial such as a trinomial. The control unit scales with the logarithm of the block since an LFSR of r flip flops may be used to count through $2^r - 1$ states. This explains why the reduction is not exactly linear. The critical timing path in all implementations is from the control logic to the shift register.

6.3 WH with Toeplitz

Table 6.4 shows the power consumptions of three implementations of WH. The first one is the standard implementation of WH with a block size of $w = 64$. The other two implementations are utilizing the multi-hashing technique with $t = 2$ and 4, and with block sizes of $w = 16$ and 32, respectively. The figures given in Table 6.4 represent the power/energy consumptions of the three hash algorithms for processing the same amount of input data (i.e. 64 bits).

In the table we observe that both the dynamic and the leakage power consumptions decrease almost linearly with increasing multi-hash iteration count t. We observe the same behavior for all frequencies. On the other hand the energy consumption remains about the same regardless of multi-hashing and only increases with decreasing operating frequency. Also notice that the leakage power remains the same and it becomes the limiting factor at lower frequencies. One way to reduce the dynamic power consumption is to lower the operating frequency. However, this increases the energy consumption as the leakage power is now consumed over a longer period of time.

Table 6.4: Comparison of Power and Energy Consumption

Frequency	**Design**	P_{Dyn}	P_{Leak}	P	E
		μW	μW	μW	nJ
100 MHz	WH-64	452.3	9.36	461.7	0.30
	WH-32	217.5	4.81	222.3	0.28
	WH-16	126.2	2.32	128.6	0.33
500 kHz	WH-64	2.261	9.36	11.62	1.49
	WH-32	1.087	4.81	5.90	1.51
	WH-16	0.631	2.32	2.95	1.51
1 kHz	WH-64	4.523	9.36	9.37	599.5
	WH-32	2.175	4.81	4.82	616.4
	WH-16	1.262	2.32	2.31	592.9

As evident from the table using the Toeplitz approach it is possible to reduce the power consumed to hash w bits of data. We next analyze the dependency of power and energy on the block size, the operating frequency, and the multi-hashing iteration count. As a first step we define w as a constant block size of 64 bits. The

Toeplitz count is t. In order to achieve the same security for an implementation with a block size of $\frac{w}{t}$ the result has to be hashed t times. The effective block length becomes $w' = \frac{w}{t}$. This approach reduces the power consumed to hash a block of w bits independently of the operating frequency as

$$\begin{aligned} P'_{Dyn} &\propto f\,w' = f\frac{w}{t} \quad , \quad \text{and} \\ P'_{Leak} &\propto w' = \frac{w}{t} \ . \end{aligned}$$

The total power consumption is found as

$$P' \propto \frac{w}{t}(cf + 1)$$

where c is a fixed constant factor. This is in line with what we have observed in Table 6.4: The total power consumption is reduced by a factor of t. This improvement does not come for free. Since we have now t blocks of length $\frac{w}{t}$, where each will be hashed t times, it will take t times longer to compute the hash of w bits of data: $T' = t\,T = t\frac{w}{f}$. However, the energy remains unaffected:

$$E' = P'\,T' \propto w^2\left(c + \frac{1}{f}\right) \ .$$

Figure 6.1 shows how the power consumption of a circuit depends on its area and the clock speed. The graph is extrapolated from simulation data at 100 MHz. It shows clearly that at low frequencies the power consumption scales linearly with the area, i.e. the leakage power is the dominant part. At higher frequencies the dynamic power takes over. The dynamic power consumption scales linearly with the frequency. Note that the frequency axis in Figure 6.1 is logarithmic and only the powers of ten are shown.

The energy consumption is shown in Figure 6.2. The axes have a different orientation than in Figure 6.1 such that the frequency is decreasing towards the right and the area is decreasing towards the left. The frequency axis in Figure 6.2 is in logarithmic scale. Figure 6.2 demonstrates that the energy consumption decreases linearly with increasing frequency. However, the energy consumption is independent of the area. This allows us to reduce the circuit size, i.e. increase the Toeplitz parameter t, without any penalty on the energy consumption. Reducing the circuit size decreases the leakage power and at low frequencies this has a big impact as shown in Figure 6.1. It is now possible to increase the frequency to a level such that the power consumption is the same as it was before reducing the area. Looking back into Figure 6.2, we can see that the energy consumption is reduced while the power consumption remained the same. This is a powerful tool for optimizing this hash function with respect to specific energy and power consumption requirements.

Equalizing Runtime We have demonstrated that the Toeplitz construction provides a drastic t-fold reduction in power consumption and circuit size at the price of t-times slower hash computation. In order to maintan the runtime one may decide to increase the operating frequency t times: $f'' = f\,t$. In this arrangement the dynamic

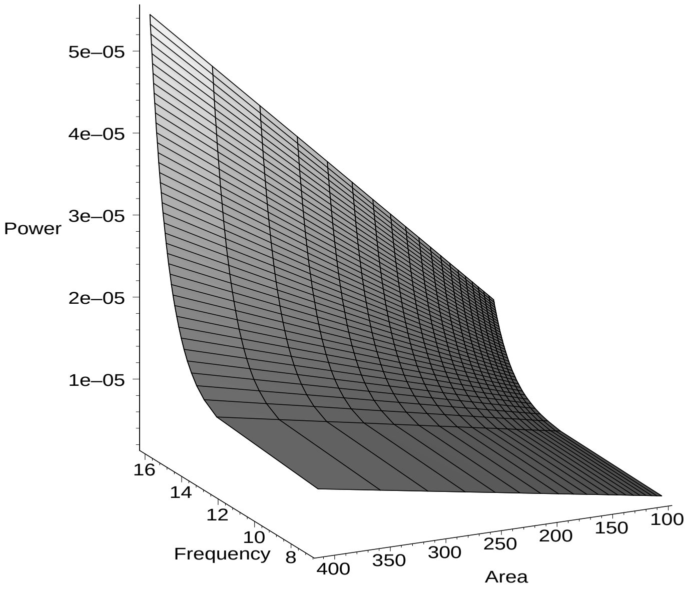

Figure 6.1: Power Consumption

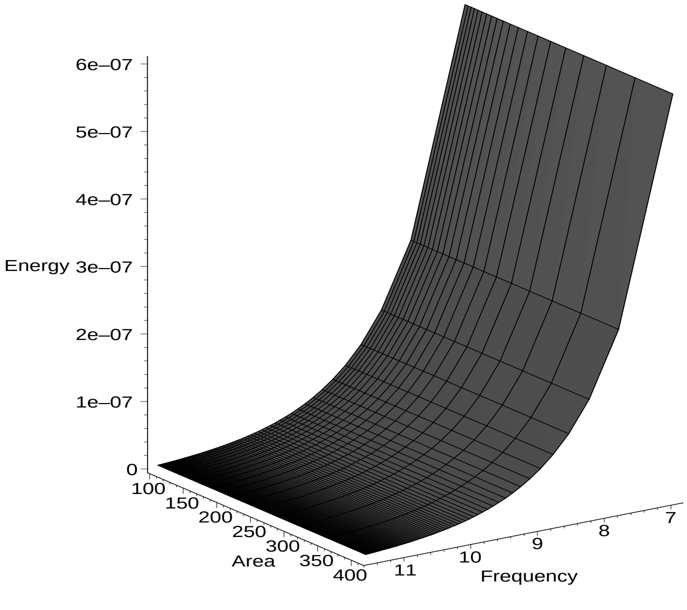

Figure 6.2: Energy Consumption

power consumption does not depend on t anymore, only the leakage power does.

$$f'' = f\,t \qquad T'' = T \propto \frac{w}{f}$$

$$P''_{Dyn} \propto f''\,w' = f\,w \qquad P''_{Leak} = P'_{Leak} \propto \frac{w}{t}$$

$$P'' \propto w\left(cf + \frac{1}{t}\right) \qquad E'' \propto w^2\left(c + \frac{1}{t\,f}\right)$$

The most important result of this is that at low frequencies (i.e. $P''_{Dyn} \ll P''_{Leak}$) the total power consumption as well as the energy consumption scales with the Toeplitz parameter t.

$$\text{for low frequencies} : E'' \propto \frac{1}{t}\frac{w^2}{f} \qquad P'' \propto \frac{1}{t}w$$

$$\text{for high frequencies} : E'' \propto w^2 \qquad P'' \propto w\,f$$

Table 6.5 shows that the energy needed to compute the hash of a 128 bit data block can be reduced without affecting the runtime. The dynamic power consumption remains roughly constant as time increases, but the leakage power consumption is reduced. Note that the header of the table specifies the frequency for WH-64 only. The frequency for WH-32 is twice higher ($t = 2$) and for WH-16 four times higher ($t = 4$).

The only way to reduce the leakage power of a circuit, aside from using a different technology, is to reduce the circuit size. Multiple hashing enables us to reduce the circuit size while maintaining the security level. The amount of additional key material is reduced through the Toeplitz approach so that this becomes a viable solution.

Table 6.5: Comparison of Power and Energy Consumption with $f' = f\,t$

Frequency	**Design**	P_{Dyn} μW	P_{Leak} μW	P μW	E nJ
100 MHz	WH-64	452.3	9.36	461.7	0.30
	WH-32	435.5	4.81	440.0	0.28
	WH-16	505.0	2.32	507.3	0.32
500 kHz	WH-64	2.261	9.36	11.62	1.49
	WH-32	2.175	4.81	7.00	0.89
	WH-16	2.525	2.32	4.84	0.62
1 kHz	WH-64	4.523	9.36	9.37	599.5
	WH-32	4.350	4.81	4.82	308.4
	WH-16	5.050	2.32	2.32	148.5

Table 6.5 shows that at 500 kHz we can reduce the power and energy consumptions by more than half and still compute the hash with the same security and in the same amount of time.

Chapter 7

Conclusions

Our main motivation for starting this research was to prove that universal hash functions can be employed to provide provable security in ultra-low-power applications such as next generation sensor networks. More specifically, hardware implementations of universal hash functions with an emphasis on low-power and reasonable execution speed are considered. We implemented NH (the underlying hash function of UMAC) for the first time in hardware and presented its simulation results in comparison to those of our newly proposed hash functions: PH, PR and WH.

The first hash function we propose, i.e. PH, produces a hash of length $2w$ and is shown to be 2^{-w}-almost universal. The other two hash functions, i.e. PR and WH, reach optimality and are shown to be universal hash functions with a much shorter hash length of w. Since their combinatorial properties are mathematically proven, there is no need for making cryptographic hardness assumptions and using a safety margin in practical implementations. In addition, these schemes are simple enough to allow for efficient constructions.

To our knowledge the proposed hash functions are the first ones specifically designed for efficient hardware implementations. Designing the new algorithms with efficiency guidelines in mind and applying optimization techniques, we achieved drastic power savings of up to 59% and speedup of up to 7.4 times over NH. Note that the speed improvement and the power reduction are accomplished simultaneously. We also observed that at lower operating frequencies the leakage power becomes the dominant part in the overall power consumption. The only way to reduce the leakage power is to reduce the circuit size. Therefore, we applied multi-hashing integrated with the Toeplitz approach to our hash function WH resulting in the designs WH-32 and WH-16. Without sacrificing any security we achieved drastic power savings of up to 90% over NH and reduced the circuit size by more than 90% to less than 500 gates at the expense of a very slight increase in the amount of key material.

We presented a powerful method for optimizing WH with respect to specific energy and power consumption requirements. We have shown that with the introduction of multi-hashing (t times) together with the Toeplitz approach the circuit size and the power consumption is reduced by a factor of t while it takes t times longer to compute the hash. Therefore the energy consumption stays about the same. On the other hand the operating frequency may be increased t times to achieve the hash without increasing the runtime. Then the dynamic power consumption is increased t-fold, however, the leakage power is not affected. Hence, at low frequencies (i.e. $P_{Dyn} \ll P_{Leak}$) the total power consumption as well as the energy consumptions decrease linearly with increasing parameter t. This is a powerful technique to decrease the circuit size, and the power and energy consumptions simultaneously while maintaining the hashing speed. The only limiting factor is the maximum operating frequency.

Note that our implementation of WH-16 consumes only $2.95\,\mu$W at 500 kHz and uses only 460 gates. It could therefore be integrated into a self-powered device. This enables the use of hash functions in ultra-low-power applications such as “Smart Dust” motes, RFIDs, and Piconet nodes. By virtue of the security and implementation features mentioned above, we believe that the proposed universal hash function together with the Toeplitz approach will fill an important gap in cryptographic hardware applications.

7.1 Future Research

We believe that ultra-low-power cryptographic hardware applications will gain more importance in the near future in accordance with the advances in ubiquitous computing. In particular, the research in universal hashing for various cryptographic hardware applications is still at an early stage. Possible directions to future research on this topic are listed below:

1. Since cryptographic hardware has to work in conjunction with other units (e.g. clock generator) of a full system, integration of universal hash function implementations with the entire chip (such as an RFID chip) will have to be studied further.

2. Circuit area and delay characteristics of universal hash function implementations may be improved utilizing the repetitive nature of Toeplitz method (described in Chapter 4). In this method the same message is hashed multiple times with the shifted (and slightly modified) versions of the same key. Therefore the same circuit can be employed with some correction terms (gates) in order to evaluate each individual hash, which could shrink down the circuit considerably.

3. [NP99] shows how universal hash functions can be combined in different ways in order to increase their domains, reduce collision probability, or decrease the range. This way it may be possible to start with fairly simple universal hash functions and by combining them properly to come up with complicated schemes with desired properties. This method may save us in terms of analysis since it could be much more time consuming (or even infeasible) to mathematically analyze the security of such a scheme without breaking it apart.

4. Universal hash functions can also be employed in *true random number generators (TRNGs)* as *randomness extractors* — a function which, when applied to the source, produces a result that is statistically close to the uniform distribution [BST03].

Bibliography

[AC98] Rajeevan Amirtharajah and Anantha P. Chandrakasan. Self-powered signal processing using vibration-based power generation. *IEEE Journal of Solid-State Circuits*, 33(5):687–695, May 1998.

[BCE+97] Frazer Bennett, David Clarke, Joseph B. Evans, Andy Hopper, Alan Jones, and David Leask. Piconet: Embedded mobile networking. *IEEE Personal Communications*, 4(5):8–15, Oct 1997.

[BHK+99] J. Black, S. Halevi, H. Krawczyk, T. Krovetz, and P. Rogaway. UMAC: Fast and secure message authentication. In *Advances in Cryptology - CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 216–233. Springer-Verlag, 1999.

[BKR94] M. Bellare, J Killian, and P. Rogaway. The security of cipher block chaining. In *Advances in Cryptology - Crypto '94*, pages 341–358. Springer-Verlag, 1994.

[Bra83] G. Brassard. On computationally secure authentication tags requiring short secret shared keys. In D. Chaum, R. L. Rivest, and A. T. Sherman, editors, *Advances in Cryptology - CRYPTO '82*, Lecture Notes in Computer Science, pages 79–86, New York, 1983. Springer-Verlag.

[BST03] B. Barak, R. Shaltiel, and E. Tromer. True random number generators secure in a changing environment. In *Cryptographic Hardware and Embedded Systems - CHES 2003*, pages 166–180. Springer-Verlag, 2003.

[CKM00] David W. Carman, Peter S. Kruus, and Brian J. Matt. Constraints and approaches for distributed sensor network security. Technical report, NAI Labs, Secutity Research Division, Glenwood, MD, Sep 2000.

[CW78] J. L. Carter and M. Wegman. Universal classes of hash functions. *Journal of Computer and System Sciences*, 18:143–154, 1978.

[CW81] J.L. Carter and M. Wegman. New hash functions and their use in authentication and set equality. *Journal of Computer and System Sciences*, 22:265–279, 1981.

[DH76] W. Diffie and M. E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, IT-22:644–654, 1976.

[DM95] Sriniwas Devadas and Sharad Malik. A survey of optimization techniques targeting low power vlsi circuits. In *Proceedings of the 32nd ACM/IEEE Conference on Design Automation*, pages 242–247, 1995.

[HHR98] K. M. Heal, M. L. Hansen, and K. M. Rickard. *Maple V Learning Guide*. Springer Verlag, New York, 1998.

[HK97] S. Halevi and H. Krawczyk. MMH: Software message authentication in the gbit/second rates. In *4th Workshop on Fast Software Encryption*, volume 1267 of *Lecture Notes in Computer Science*, pages 172–189. Springer, 1997.

[Kra94] H. Krawczyk. LFSR-based hashing and authentication. In *Advances in Cryptology - Crypto'94*, volume 839 of *Lecture Notes in Computer Science*, pages 129–139. Springer-Verlag, 1994.

[Kra95] H. Krawczyk. New hash functions for message authentication. In *EUROCRYPT'95*, volume 921 of *Lecture Notes in Computer Science*, pages 301–310. Springer-Verlag, 1995.

[KRP99] J. M. Kahn, Katz R. H., and K. S. J. Pister. Next century challenges: mobile networking for "smart dust". In *Proceedings of the fifth annual ACM/IEEE international conference on Mobile computing and networking*, pages 271–278. ACM, 1999.

[LC02] Philip Levis and David Culler. Mat: a tiny virtual machine for sensor networks. In *Proceedings of the 10th international conference on architectural support for programming languages and operating systems (ASPLOS-X)*, pages 85–95, San Jose, California, 2002. ACM Press.

[LRW92] X. Lai, R.A. Rueppel, and J. Woollven. A fast cryptographic checksum algorithm based on stream ciphers. In *Advances in Cryptology - Auscrypt '92*, pages 339–348. Springer-Verlag, 1992.

[MMMA$^+$01] Scott Meininger, Jose Oscar Mur-Miranda, Rajeevan Amirtharajah, Anantha P. Chandrakasan, and Jeffrey H. Lang. Vibration-to-electric energy conversion. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 9(1):64–76, Feb 2001.

[MNT90] Y. Mansour, N. Nissan, and P. Tiwari. The computational complexity of universal hashing. In *22nd Annual ACM Symposium on Theory of Computing*, pages 235–243. ACM Press, 1990.

[Nat02] National Institute of Standards and Technology (NIST), FIPS Publication 180-2. *Secure Hash Standard (SHS)*, Aug 2002. http://csrc.nist.gov/publications/fips/fips180-2/fips180-2.pdf.

[NP99] W. Nevelsteen and B. Preneel. Software performance of universal hash functions. In *EUROCRYPT'99*, volume 1592 of *Lecture Notes in Computer Science*, pages 24–41, Berlin, 1999. Springer-Verlag.

[Par00] Behrooz Parhami. *Computer Arithmetic: Algorithms and Hardware Designs*. Oxford University Press, 2000.

[PR99] M. Etzel S. Patel and Z. Ramzan. SQUARE HASH: Fast message authentication via optimized universal hash functions. In M. Wiener, editor, *Advances in Cryptology - CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 234–251, New York, 1999. Springer-Verlag.

[PST$^+$02] A. Perrig, R. Szewczyk, J. D. Tygar, V. Wen, and D. E. Culler. SPINS: security protocols for sensor networks. *Wireless Networks*, 8(5):521–534, Sep 2002.

[RFB94] M.V. Ramakrishna, E. Fu, and E. Bahcekapili. A performance study of hashing functions for hardware applications. In *Proceedings of the ICCT '94 International Conference on Computing and Information*, pages 1621–1636, 1994.

[Riv92] R. Rivest. The MD5 message-digest algorithm. RFC 1321, MIT Laboratory for Computer Science and RSA Data Security Inc., Apr 1992.

[Rog95a] P. Rogaway. Bucket hashing and its application to fast message autheti-cation. In D. Coppersmith, editor, *Proceedings Crypto '95*, volume 963 of *Lecture Notes in Computer Science LNCS*, pages 29–42. Springer-Verlag, 1995.

[Rog95b] P. Rogaway. Bucket hashing and its applications to fast message authentication. In *Advances in Cryptology - CRYPTO '95*, volume 963 of *Lecture Notes in Computer Science*, pages 313–328, New York, 1995. Springer-Verlag.

[RP96] Jan M. Rabaey and Massoud Pedram. *Low Power Design Methodologies.* Kluwer Academic Publishers, Norwell, Massachusetts, 1996.

[SBA00] Sanjay Sarma, David L. Brock, and Kevin Ashton. The networked physical world - proposals for engineering the next generation of computing, commerce & automatic identification. White paper, MIT: Auto-ID Center, Oct 2000.

[Sho96] V. Shoup. On fast and provably secure message authentication based on universal hashing. In *Advances in Cryptology - CRYPTO '96*, volume 1109 of *Lecture Notes in Computer Science*, pages 74–85, New York, 1996. Springer-Verlag.

[Sim92] G. J. Simmons, editor. *Contemporary Cryptology.* IEEE Press, 1992.

[Sti95] D. R. Stinson. *Cryptography - Theory and Practice*. CRC Press, Boca Raton, second edition edition, 1995.

[Syn02a] Synopsys Inc. *Design Compiler User Guide*, version 2002.05 edition, Jun 2002.

[Syn02b] Synopsys Inc. *Power Compiler User Guide*, release 2002.05 edition, May 2002.