Cryptography for Ultra-Low Power Devices

by
Jens-Peter Kaps

A Dissertation
Submitted to the Faculty
of the
WORCESTER POLYTECHNIC INSTITUTE
In partial fulfillment of the requirements for the
Degree of Doctor of Philosopy
in
Electrical Engineering

May, 2006

Approved:

Abstract

Ubiquitous computing describes the notion that computing devices will be everywhere: clothing, walls and floors of buildings, cars, forests, deserts, etc. Ubiquitous computing is becoming a reality: RFIDs are currently being introduced into the supply chain. Wireless distributed sensor networks (WSN) are already being used to monitor wildlife and to track military targets. Many more applications are being envisioned. For most of these applications some level of security is of utmost importance. Common to WSN and RFIDs are their severely limited power resources, which classify them as ultra-low power devices.

Early sensor nodes used simple 8-bit microprocessors to implement basic communication, sensing and computing services. Security was an afterthought. The main power consumer is the RF-transceiver, or radio for short. In the past years specialized hardware for low-data rate and low-power radios has been developed. The new bottleneck are security services which employ computationally intensive cryptographic operations. Customized hardware implementations hold the promise of enabling security for severely power constrained devices.

Most research groups are concerned with developing secure wireless communication protocols, others with designing efficient software implementations of cryptographic algorithms. There has not been a comprehensive study on hardware implementations of cryptographic algorithms tailored for ultra-low power applications. The goal of this dissertation is to develop a suite of cryptographic functions for authentication, encryption and integrity that is specifically fashioned to the needs of ultra-low power devices.

This dissertation gives an introduction to the specific problems that security engineers face when they try to solve the seemingly contradictory challenge of providing lightweight cryptographic services that can perform on ultra-low power devices and shows an overview of our current work and its future direction.

Preface

This dissertation describes the research I conducted at the Worcester Polytechnic Institute (WPI) in the past four years. I hope that the work presented here serves as a tutorial to the specific problems of providing cryptography for ultra-low power devices and that it lays the foundation for future research in this area.

This work would not have been possible without the support and help of many people here at WPI and from outside. First of all I want to thank my advisor Prof. Berk Sunar who agreed to be my mentor while I was crossing the United States on a bicycle in order to attend the Cryptographic Hardware and Embedded Systems (CHES) in San Francisco. He was always available and provided me with guidance, advice, support and inspiration.

I am grateful to my dissertation committee Prof. Wayne Burleson, Prof. Wenjing Lou, and Prof. John McNeill for their support. Prof. Burleson opened my eyes to the low level VLSI design issues which I now propose for future work. Prof. Lou gave me insights into the workings of security protocols. Prof. McNeill provided me with many valuable comments and suggestions.

I would particularly like to thank my co-conspirators Gunnar Gaubatz, Kaan Yüksel, and Erdinç Öztürk. Gunnar and I did not only work together on many papers, live in the same lab and the same house, he is also a very good friend. The work on NH and its derivatives would not have been possible without Kaan who researched universal hash functions and formulated the mathematical descriptions and proofs. Erdinç implemented the ECC point multiplication shown in Chapter 5. Thanks to Prof. Martin for his invaluable help with the proofs for the universal hash function families presented in Chapter 4.

Thanks go also to Prof. Fred Looft, our department head, who let me experiment with my teaching skills on four students during a summer course and then hired me as an Instructor. This experience gave me unique insights into what it means to be a professor and I learned many valuable lessons. Nothing would work in the department without the help of the administrative assistants Cathy Emmerton, Brenda McDonald, and Colleen Sweeney. But I want to thank them for an even more important service: providing free sweets in the department office so I can keep my sugar level high.

I would also like to thank Prof. Christof Paar, who was my advisor for my Master's thesis, for his continued friendship and many interesting discussions.

There are also several students I would like to thank. Amongst them are David Holl, Kui Ren, and Benjamin Woodacre who suffered through my several presentation rehearsals and with whom I had many fruitful discussions. Thanks to Shiwangi Despande, whom I'm indebted to, and my lab mates Selçuk Baktır, Deniz Karakoyunlu, and Ghaith Hammouri.

I also want to thank the National Science Foundation for partially funding this research through the Grants No. ANI-0112889 and ANI-0133297.

Last, but not least, I want to thank my fiancée Alpna Saini for all her love and support and my parents for their continuous encouragement.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Computing technology is reaching every corner of our lives. Mobile communication, personal computation (e.g. personal digital assistants: PDAs), and portable navigation devices are just a few examples of the most commonly known applications. Recent advances in ultra-low-power technology enabled the development of even smaller, more mobile, autonomous devices. Wireless Sensor Networks (WSN) and RFIDs [126] are a few examples of this trend.

The power available to WSN sensor nodes and RFID tags is orders of magnitude less than what common battery powered devices consume. Batteries for these devices are tiny and can supply $10\,\mu$W for only one day [65]. Moreover, some of these technologies collect energy from environmental sources, such as light, heat, noise, or vibration using *power scavengers* which produce between $1\,\mu$W and $500\,\mu$W. Scavengers and RFID reader fields do not produce enough energy to support even the simplest general purpose low power CPUs which are currently being used in Sensor Networks. This defines the domain of ultra-low power. Table 1.1 shows how "ultra-low power" fits into the power spectrum of computing devices.

Table 1.1: Power Spectrum

Application	Power Source	Power Range		
Desktop Computer	Power Grid	150 W	–	500 W
Laptops	High Capacity Battery	10 W	–	120 W
Palm size devices				
Cell Phones	Battery	100 mW	–	10 W
Embedded Systems				
Wireless Sensors	Small Batteries	1 mW	–	100 mW
Smart Dust, RFID	Energy Scavenging	1 μW	–	500 μW

Another commonality to WSN nodes and RFID tags is that they communicate wirelessly, hence their communication is easy to intercept and to tamper with. Due to the sensitive nature of many of the anticipated applications a certain level of security is crucial. However, these severe power constraints make designing cryptographic systems especially difficult. In the following sections we explore the technology behind RFID and WSN and show their main applications. We identify the security needs associated with these applications and have a look at what work has been done previously to address these needs.

1.1.1 Wireless Sensor Networks

Wireless sensor networks (WSN) [27], [28], [29] enable monitoring of natural and man-made environments at a never before seen level of granularity. This macroscopical view is achieved by placing 10's to 1000's of small wireless sensors, so called sensor nodes or motes, in the target area to sense fields and forces. For this, each sensor node contains sensors, limited data processing capabilities, data and energy storage, and a wireless transceiver enabling them to form a network and work together as an ensemble. In order to be mass deployable, the sensor nodes must be inexpensive, expendable, easy to deploy, use, and maintain.

These features make WSN very flexible and opened up a wide range of applications. An overview of the history of sensor networks can be found in [24]. Wireless sensor networks started out in the military with sensor nodes so big as to till the bed of a truck. The first reported deployments of small, pager sized nodes were [86] for environmental monitoring and [139] for habitat monitoring. Examples are monitoring birds on Great Duck Island on the coast of Maine [110], [82] and collecting microclimate data in the James San Jacinto Mountains Reserve [39], [21]. There are also several military applications like target tracking [17], [7], detecting radiation [16], biological, and chemical weapons, and localization of shooters in urban terrain [85]. Most of these applications require the nodes to operate unattended for a long period of time which is limited by their energy source, usually a battery. In order to minimize energy consumption the nodes have a low duty cycle, i.e. most of the time they are turned off. The range of their radio transmitters for wireless data transfer is limited to conserve power, hence they can only communicate directly with nodes in close proximity. They establish a routing tree with the base station at its root. The base station collects the data from the sensors and communicates with the outside world. It is assumed to have sufficient power for all computations and communications with the nodes and the outside world [108].

Piconet [13] was an early general-purpose, low-power ad hoc radio network by the University of Cambridge. The "Smart Dust" project at the University of California, Berkeley [65] set out to develop sensor nodes of 1mm^3 in size. Their early studies even showed designs for flying motes [145] which probably inspired the book "Prey" by Michael Crichton [26]. These early Smart Dust motes used a steered laser beam for communication [13], [146], [147], however battery powered nodes with wireless radio frequency transmitters have become standard [47]. Current sensor platforms can be divided into four classes [54]: Gateway, high bandwidth sensing, generic sensing, and specialized sensing. Table 1.2 shows example motes for each class and their power consumption. The power for sending and receiving is consumed by the RF-transmitter in addition to the power consumed by the CPU unless noted otherwise. The gateway class provides the connection from a sensor network to traditional networks like Ethernet, 802.11 wireless, etc. and functions as a base station. High bandwidth sensing is used for video and sound which requires significant processing power and a high bandwidth radio. BTmode and Imote [70] are currently the most popular choices in this class. The most common platform is generic sensing for which the *Mica 2* and Telos [111] motes from Berkely are examples. These devices are the size of a modern pager, use two AA batteries and consumes less than 30 mW when active. The radio consumes an additional 40 mW when sending and about the same amount when receiving. While sleeping, these mote consumes less than 45 μW. Sensor node platforms for generic sensing and for specialized sensing use simple 8-bit

Table 1.2: Sensor Node Platforms

Platform	Example	Processor	Memory ROM [kB]	Memory RAM [kB]	Processor Active [mW]	Processor Sleep [μW]	Radio Send [mW]	Radio Receive [μW]
Specialized Sensing	Spec	Custom 8-bit	0	3	1.5		1	900
	Mica2Dot	ATmega128L	128	512	24	< 45	42	29
Generic Sensing	Mica2	ATmega128L	128	512	24	< 45	42	29
	Telos	TI MSP430	48	1024	6	< 15	35	38
High Bandwidth	Imote	ARM7TDMI	512	64	195	300	incl. in Power	
	BTnode	ATmega128L	128	244	39.6	9.9	66	50
Gateway	Stargate	Intel PXA255	32,000	64,000	$< 2,500$			

general purpose processors. Kahn describes in [65] that tiny batteries for these devices can supply 10 μW for only one day. Hence, the nodes have to operate on a low duty cycle. Mica2Dot is only as big as a Quarter coin but its power consumption is equivalent to that of the Mica 2 mote however it provides less extension possibilities and therefore fits into the specialized sensing category. When powered with a small

3V battery its lifetime will be significantly shorter than that of the Mica2 due to the lower capacity of the battery. The "Smart Dust" project also developed the *Spec* (of dust) mote [56] which is only $5\,\mathrm{mm}^3$ in size. The overall dimensions of the motes depends mainly on the size of the batteries. Advances in battery technology and development of other power sources [47] will make smaller motes possible.

The current generation of wireless sensor nodes is relying on batteries as its source of power. The limited lifetime of batteries, however, significantly impedes the usefulness of such devices since maintenance access would become necessary whenever the battery is depleted. Sensor nodes could only be deployed in accessible areas and not, for example, be dropped off airplanes over enemy territory. Furthermore, the intention of having large amounts of tiny nodes scattered over a large area would render maintenance impractical.

We envision that the next generation sensor nodes will operate without batteries. Instead they will harvest energy from ambient sources in their environment such as light, heat, and vibration. The notion of removing the battery and having self-powered computing devices opens the door to a wealth of new applications, not just for wireless sensor nodes but for the whole field of ubiquitous computing. Computing devices can then be embedded in walls, in concrete, or placed in inaccessible locations.

Energy scavengers are devices that harvest energy from environmental sources and convert it into electric power. This energy is stored in a capacitor and can be used to power the sensor node either continuously, for small amounts of power, or in intervals if the demand is higher. Autonomous nodes which use scavengers are called *self-powered*. An implementation of a signal processing unit powered by a large scavenger device that can generate up to $400\,\mu\mathrm{W}$ is described in [5]. Commercially available large scavengers can produce up to $3\,\mathrm{mW}$ at a cost of \$30 per unit [25]. Newer scavengers are based on micro-electromechanical systems and have been integrated on chip which reduces size and cost. They are currently capable of producing up to

8 μW of power [89] relying solely on ambient vibration. It is expected that future MEMS-based scavengers will be able to deliver power up to $50\mu W$ continuously.

1.1.2 Radio Frequency Identifiers

Radio Frequency Identifiers (RFID) systems enable wireless identification and tracking of items. For this, small, inexpensive electronic devices, so called RFID tags or transponders, are affixed to the items that are to be identified. The tags contain at a minimum a wireless RF transceiver and a unique identifier (ID). Tags get queried by readers which are more complex and usually connected to a backend system via some form of network. Upon being queried, the tags respond with their IDs. The readers are capable to query multiple tagged items at once and distinguish between each one of them.

The first RFID like systems were simple electronic article surveillance (EAS) systems developed in the late 1960s [77]. In this application the tags did not transmit an ID they just indicated their presence. In the 1970s researchers were able to construct tags that respond with an ID number. This opened up a wealth of commercial applications starting with livestock tracking [1] using implantable devices around 1990 to electronic toll collection for highways, container tracking, car anti-theft, employee badges, and wireless electronic payment for gasoline and other goods just to name a few. Several different and incompatible types of identifiers and technologies are used, depending on application and device vendor. In 1999, the AutoID center at the Massachusetts Institute of Technology was founded to guide and standardize the development of RFID technology [126] which resulted in the adoption of the Electronic Product Code (EPC) as an international standard. The idea behind the EPC is to replace the ubiquitous barcodes, the Universal Product Codes (UPS), which can be scanned by a laser, with simple, cheap, 5 US cent a piece RFID tags that can be read without line of sight. One example of how RFIDs are moving into the mainstream consumer market is the news that Walmart is planning to track pallets and cases from its suppliers using RFIDs [2].

The most basic RFID tags, which are also the most common, contain a radio frequency transponder and a read-only memory chip that contains a unique identifier. This identifier follows either a proprietary scheme or it is the standardized electronic product code (EPC), also called Global Unique Identifier (GUID). We distinguish between active and passive tags. Active tags carry a small battery while the cheaper and much more common passive tags receive their power from the reader. The reader emits an electric field while querying the tags, which also powers the tags. The amount of power a tag receives depends on the field's intensity which is governed by national and international regulations. Only about 20 μW are available for the digital part of an RFID tag.

More sophisticated tags have more functionality than just responding to a reader with their ID [144]. Some new tags have writable memory so that a reader can store information on the tag. This can be used to maintain a log of a tags state during each read right on the tag. Other new tags have embedded sensors, blurring the line with sensor nodes. These sensors can not consume any power while the tag is not exposed to the RF field of a reader. However simple bacteria and temperature sensors have been developed that change their electrical properties depending on their environmental conditions and maintain that state. One example are temperature sensors that detect whether a food item has become to warm, or bacterial sensors measuring bacterial contamination in food and indicating once a certain threshold has been exceeded. When queried by a reader, these sensor RFID tags respond with their ID and the state of the sensor.

Nowadays all functions, from RF transponder to ID memory and also many of the new additional functions like sensing and re-writable memory, are integrated into one tiny custom chip. Researchers envision that wireless sensor nodes will make a similar progression as RFIDs did over the past 40 years and soon also be the size of a grain of rice. However the size of the tags is limited be the size requirements for the antenna.

1.1.3 Security Concerns

Wireless sensor networks developed for researchers to study natural environments and monitor animal habitats had no need for any form of security. The data gathered by the sensor nodes was public and there was no benefit for an attacker in tampering with the operation of the network. Radio frequency identification tags were originally designed for monitoring livestock. The security concerns for this application were also minimal. However, with the increasing number of applications for both technologies the security concerns are becoming the bottle neck for widespread deployment.

Wireless Sensor Networks A general overview of security issues for general wireless networks can be found in [132]. Ad-hoc sensor networks (ASN) are a special form of wireless sensor networks (WSN) in that the nodes are highly mobile. In some cases the membership in the network can change, new devices can be added, others retired, and some might be moved to a different network. A good introduction to the security issues that WSN face can be found in [109] and [61]. Here are some examples for WSN that we refer to throughout this dissertation and the security concerns they raise.

WSN Example 1: WSN nodes can be used to monitor the mechanical stress of a bridge. Any crack in the concrete, deformation of the steel rods will be picked up by the sensors and relayed to a monitoring facility. This information is used to generate an up to the minute status of the bridge. In the event of a major accident or an earthquake, the operators can immediately asses the damage and determine if the bridge is still save or has to be closed. This type of monitoring can be achieved by mixing thousands of maintenance free, vibration scavenger powered, grain of rice sized, inexpensive sensor nodes into the concrete when the bridge is constructed. Once in place, the sensor nodes will form a network and relay their measurement data to one or more base stations. These sensor nodes have limited transmission power and range. Therefore, it would be easy for an attacker to send fake information using a high powered device like a laptop and fool the authorities to close the bridge.

WSN Example 2: WSN nodes can be used for military purposes, for example to detect and gain as much information as possible about enemy movements, explosions, and other phenomena of interest. This information is relayed to mobile command posts where it helps the commanders to make decisions about troop movements, calls for air support, etc. The sensors can either be dropped off an aircraft over enemy territory after which they lie stationary on the ground, or they are carried by each soldier and vehicle and therefore form a mobile ASN. In this scenario it is of utmost importance to prevent the enemy from listening to the transmitted data. In addition to this, it is very likely that some nodes fail and stop operating and, even worse, that some nodes get captured by the enemy.

Radio Frequency Identification Devices Security aspects of RFIDs have been examined in[151], [127], and [150]. We are presenting two popular applications for RFID tags and highlight their security concerns.

RFID Example 1: RFID tags with Electronic Product Codes (EPC) are going to replace the ubiquitous optical barcodes which are showing the universal product code (UPC), on consumer items. The wireless nature with no line-of-sight requirement makes RFID ideal for inventory control and fast check out. Workers in a shop only need to hold the RFID reader in front of a shelf and immediately know how many items of each product are in the shelf. At the checkout, the shoppers only have to put the goods on the belt which passes them through a reader. There is no need to search for the barcode and align it with an optical scanner. This will speed up the checkout process. However, this raises security concerns. If the tags do not get deactivated at the time of checkout, it is possible to track individuals by the unique combination of tags in their clothing. The book Spychips[1][3] elaborates on the threats to privacy that RFIDs pose. In spite of this, a consumer might want to keep the RFID tags active due to other benefits they offer. A RFID enabled washing machine can warn the user if it detects red socks in a load of white shirts. In order to make tracking more difficult but enabling some of the benefits a deletion of unique identifier but not the products properties might be sufficient. That however raises the question on which reader is allowed to erase information of the tag.

RFID Example 2: Wireless electronic payment e.g., Exxon Mobile Speedpass[2], is another growing application for RFID. The tag's number is linked to the users credit card information. When filling up a tank of gasoline, the user waves the RFID device in front of the fuel pump and the gasoline will get billed to users credit card. However, an attacker on the opposite fuel pump could intercept the messages with an unlicensed and therefore more powerful reader and then use this information to charge her fuel to the legitimate user. This example illustrates that it is important to protect the wirelessly transmitted data from an eavesdropper.

1.1.4 Security Services

Most of the security concerns we just discovered can be addressed by the services of availability, authenticity, integrity, confidentiality, trust setup and scalability. Cryptographic functions are the fundamental building blocks of most of these services.

[1]Spychips is a trademark of Katherine Albrecht and Liz McIntyre

[2]Speedpass is a registered trademark of the Exxon Mobile Corporation

Availability The functionality of the sensor network must be ensured in spite of denial-of-service attacks (DoS) or node failures, i.e. the network must be available to authorized parties. The countermeasures to these kind of concerns are at the network protocol level and therefore not addressed in this dissertation.

Authenticity In WSN and for RFID applications authenticity is of utmost importance. WSN Example 1 shows that an attacker can easily inject unauthenticated messages into a bridge monitoring network. RFID Example 1 raises the question on which reader is allowed to erase information on a tag. If this feature should be allowed only for a retail store, then the tag has to accept the delete commands only after it has successfully established the authenticity of the reader. The service of authenticity can be provided by message authentication codes (MAC), keyed hash functions and digital signatures.

Integrity For all of the examples above, the integrity of data that is transmitted between WSN nodes or an RFID tag and the reader must be guaranteed. Wireless data transmission for ultra-low power WSN nodes and RFID tags is very unreliable due to their low-power radio transmitters. Furthermore attackers can easily modify the data of messages in transit. Message authentication codes, keyed hash functions and digital signatures can guarantee message integrity and at the same time authenticity.

Confidentiality Keeping data secret is crucial for military applications (WSN Example 2) and electronic payment applications (RFID Example 2). The standard solution is for the sender to encrypt the data with a secret key, known only to the sender and receiver. The receiver would then decrypt the data.

Trust Setup and Scalability In order to establish trust between sensor nodes or RFID tags and the readers secret cryptographic keys have to be established between all participants. The simplest idea is to use one common shared key. However, if a node, or an RFID tag is captured, the attacker might be able to retrieve the secret key which would give access to all network traffic. The capture of a single sensor node by an adversary should not jeopardize the integrity of the entire network. Another approach is to establish a secret between each pair of communicating partners. For a network with n nodes that potentially all talk to each other it would require $n \cdot (n-1)/2$ keys. This does not scale for the thousands of nodes or millions of tags that are going to be used in the above examples. Other approaches are the use of shared keys or random key pre-distribution for which many schemes have been proposed [22], [154], [35]. Public key cryptography offers an elegant way for establishing secret keys, even though the computational complexity seems daunting. We are exploring this specific case in Chapter 5.

1.2 Previous Work

In order enable the above mentioned security services for ultra-low power devices we need to have special hardware implementations of cryptographic functions. When designing cryptographic hardware that can perform under the constrained environments of RFIDs and WSN the power consumption has to be made the first design priority. So far research has concentrated on the network specific aspects of WSN and on software implementations of cryptographic algorithms. The security aspects of distributed sensor networks have been reviewed by *NAI Labs* in [19]. 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. Only recently studies were published on special hardware implementations [38, 42, 41, 153].

The services of Authenticity and Integrity can be provided by message authentication codes (MACs). Universal hash function families, as discussed in Chapter 4, can be used to build secure MACs. 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 [116]. However, the main emphasis was on using hash functions for table organization and address translation. In an early work Krawczyk [73] proposed new hash functions from a hardware point of view. This work introduced two constructions: a CRC-based cryptographic hash function, and a construction based on Toeplitz hashing where matrix entries are generated by a Linear Shift Feedback Register (LFSR). 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 [130, 49, 123, 74, 14, 36]. For a survey see [100]. 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 (e.g. PDAs, mobile phones) space and power limitations prohibit their employment.

Complex key management and high storage requirements for multiple keys and messages put a considerable burden on the power consumption of the nodes. The use of public key cryptography would eliminate the need for complicated protocols and at the same time would also increase the security of the entire system, since only the public key of the base station would have to be embedded into the nodes. However, until recently, most publications on ultra-low power cryptography claim that public key cryptography is not feasible on for WSN nodes and RFID tags [108], [109], [61].

Many research papers [108, 78, 81, 40, 46] have been published for secret key cryptography which is used to provide confidentiality. However, most of them analyze encryption algorithms for wireless sensor networks and RFIDs exclusively with reference to speed and code size while only a few [112] address the energy consumption of software based implementations. However, the ultra-low power applications we are envisioning, do not provide enough power for running cryptographic algorithms on general purpose microprocessors. The first ultra-low power hardware implementation of a block cipher was reported in [38] and [37].

The main power consumer of a wireless sensor node is its RF-transceiver or radio for short. In the past years specialized hardware has been developed for the radios which combine low-data rate, low-power consumption, and the ability to interface directly with low-power micro controllers. We propose that in order to provide adequate security for ultra-low power applications like WSN and RFID, one has to employ a similar approach as for the radios, i.e. specialized hardware in conjunction with application specific algorithms. The main design goal is to have ultra-low power implementations. However, other factors also play a role. In spite of the advances in radio design, transmission power is still very costly compared to computations. On most WSN nodes, transmitting a single bit costs as much power as executing 1000 instructions. Hence, the transmission overhead incurred by applying security has to be kept to an absolute minimum. Furthermore, most cryptographic algorithms currently in use are designed for high performance, mostly in software on 32-bit microprocessors. On sensor nodes, or RFIDs the computation time is not very critical. It is more important to consume little power and space. Our goal is to develop a suite of cryptographic functions for authentication, encryption and integrity that is specifically fashioned to the needs of ultra-low power devices.

1.3 Thesis Outline

We started our quest for a suit of cryptographic functions for ultra-low power devices with a close look at the power dissipation of CMOS circuits and low power design methodologies in Chapter 2. We use the low power design considerations to explore current cryptographic primitives and analyze their suitability for ultra-low power implementations in Chapter 3.

The first function we implemented was the universal hash function family NH which has provable security properties. We introduce three variations on NH, each showing an improvement in power consumption, and prove that each is at least as secure as the original NH. We then show how the technique of multihashing 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 key material in Chapter 4.

These results encourage us to challenge the prevalent notion that public key cryptography is not feasible for wireless sensor notes. In Chapter 5 we present proof-of-concept implementations of three different algorithms—Rabin's Scheme, NtruEncrypt, and Elliptic Curve Cryptography—and analyze their architecture and performance according to various established metrics.

In Chapter 6 we complete our suit of cryptographic functions by exploring secret key algorithms. We present a novel ultra-low power SHA-1 design and an energy efficient AES design. Both can be used for authentication and encryption.

We mentioned in Section 1.2 that transmission power is very costly. In Chapter 7 we are analyzing the two most popular WSN protocols and present a list of fundamental security services that would be best served by using public key cryptography. This analysis is corroborated with data from our public key algorithm implementations from Chapter 5.

Chapter 8 closes this dissertations with our conclusions and recommendations for future research.

Chapter 2

Ultra-Low Power Hardware Design

In order to provide cryptographic functions for ultra-low power devices, the power consumption has to become the major design consideration. This Chapter provides an overview of the sources for power dissipation in CMOS circuits and then presents a few methods on how to limit power consumption at different levels of circuit design. A detailed description of low power design methodologies can be found in these books [114], [105] and these journal papers [12], [104], [11].

2.1 Sources of Power Dissipation

In order to design ultra-low power circuits we have to examine and understand the sources of power dissipation in CMOS devices. The following formula characterizes the power dissipation [31]:

$$P = \underbrace{\left(\frac{1}{2} \cdot C_L \cdot V_{DD}^2 + Q_{SC} \cdot V_{DD}\right) \cdot f \cdot N}_{\mathrm{P_{Dynamic}}} + \underbrace{I_{leak} \cdot V_{DD}}_{\mathrm{P_{Leakage}}} \qquad (2.1)$$

The term V_{DD} is the supply voltage. $\mathrm{P_{Dynamic}}$ represents the dynamic power dissipation caused by changes to the outputs of the CMOS circuit i.e. transitions from logic 0 to logic 1 or vice versa. The second term $\mathrm{P_{Leakage}}$ represents the static power consumption which is dissipated while the circuit is powered on. It is independent of switching activity and frequency, therefore we assume it to be constant for a certain circuit. We examine the two major sources of power dissipation in detail.

2.1.1 Dynamic Power

The dynamic power consumption consists of two parts: switching power and short circuit power. Both are depending on the switching activity, i.e. the number of gate output transitions per clock cycle N, and the operating frequency f.

$$\mathrm{P}_{\mathrm{Dynamic}} = P_{SW} + P_{SC}$$

For our discussion of the dynamic power consumption we use the simplest CMOS gate, an inverter (Figure 2.1), as a general model of any CMOS gate.

2.1.2 Switching Power

The switching power dissipation describes the power dissipated by charging and discharging the nodes capacitance C_L, it is therefore also referred to as the capacitive power dissipation. For simplification we assume that all capacitances are at the output node. In this model, the capacitance C_L consists of two parts: C_1 and C_2, as shown in Figure 2.1, where C_1 is a capacitance to ground and C_2 to V_{DD}. When the

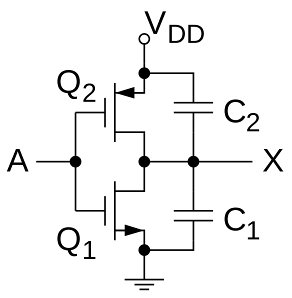

Figure 2.1: Inverter Output Capacitances

input A is at logic 0, the n-channel transistor Q_1 is open and the p-channel transistor Q_2 is closed, leading to a logic 1 at the output X. At the same time, the capacitance C_2 is emptied, shortened by Q_2 and the capacitance C_1 is charged. A change on A from logic 0 to 1 causes Q_1 to close, which will discharge C_1 and Q_2 to open, which will charge C_2. During this single transition, a charge of $C_2 \cdot V_{DD}$ was taken from the power supply. On the next input change the charge $C_1 \cdot V_{DD}$ will be taken from the power supply. Assuming that $C_L = C_1 + C_2$, we can say that after one input transition from 0 to 1 and back to 0 the inverter has taken a charge of $(C_1 + C_2)V_{DD}$ from the power supply. Therefore, we can average this for one single output transition to $\frac{1}{2} \cdot C_L \cdot V_{DD}$. The power dissipated by this is $\frac{1}{2} \cdot C_L \cdot V_{DD}^2$ which leads to Equation 2.2 for the switching power P_{SW}.

$$P_{SW} = \frac{1}{2} \cdot C_L \cdot V_{DD}^2 \cdot f \cdot N \tag{2.2}$$

2.1.3 Short Circuit Power

During an input transition of a CMOS circuit there is a short time period in which the n-channel and the p-channel transistors are both on, causing a short circuit from V_{DD} to ground. This current, I_{SC}, flows while the input voltage is between V_T and $V_{DD} - V_T$ where V_T is the threshold voltage. The amount of current is depending on the gain factor β (amplification) of a MOS transistor and input rise and fall times τ. Veendrick [142] computed the short circuit current as

$$I_{SCmean} = \frac{1}{12} \cdot \frac{\beta}{V_{DD}} \cdot (V_{DD} - 2V_T) \cdot \frac{\tau}{T} \; .$$

Using the fact, that $\frac{1}{T} = f$ we can express P_{SC} as

$$P_{SC} = \left(\frac{1}{12} \cdot \frac{\beta}{V_{DD}} \cdot (V_{DD} - 2V_T) \cdot \tau \right) \cdot V_{DD} \cdot f \cdot N \tag{2.3}$$

This approximation was improved, amongst others, by Turgis et al. in [141] with an explicit formulation for P_{SC}. They also introduced an idea to express the short-circuit power dissipation through an equivalent charge Q_{SC} carried by the short-circuit current which leads to Equation 2.4.

$$P_{SC} = Q_{SC} \cdot V_{DD} \cdot f \cdot N \tag{2.4}$$

2.1.4 Static Power

The static power consumption of a CMOS circuit is caused by the leakage currents of transistors and pn-junctions. If we go back to Figure 2.1 and assume the input is a logic 0, we know that the n-channel transistor Q_1 is open. The leakage power is given by the leakage current I_{0n} of this transistor. When the input is a logic 1 and then the power is given by the leakage current I_{0p} of the p-channel transistor Q_2. This can be expressed by Equation 2.5.

$$\mathrm{P}_{\mathrm{Leakage}} = P_S = V_{DD} \cdot \frac{I_{0n} + I_{0p}}{2} = I_{leak} \cdot V_{DD} \tag{2.5}$$

Leakage currents are increasing from one CMOS technology generation to the next due to the reduction of threshold voltage, channel length, and gate oxide thickness [124]. In future generations the leakage power is projected to become the dominant part of the overall power consumption [30][63].

2.1.5 Glitching

Glitches are unwanted transitions of a signal after an input change until the final output value is reached. This behavior is due to different arrival times of signals to a gate, called logic hazards. Figure 2.2 shows the circuit for the logic equation $Q = A\overline{B} + BC$ which exhibits a static-1 hazard. When the inputs A and C are logic 1 any change on B will cause a transition on Q. There are two paths for B to the output Q where one path contains an inverter. This causes a slightly longer delay, resulting in a glitch in the output Q. Larger, more complex circuits e.g. ripple carry

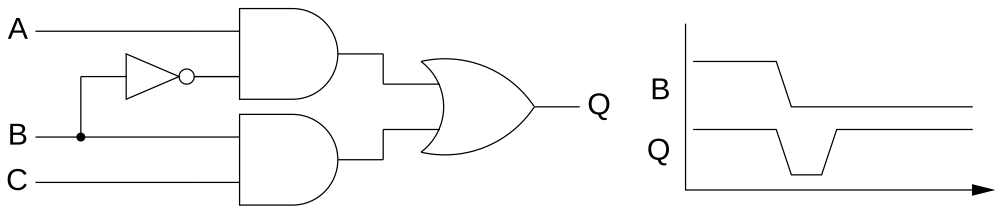

Figure 2.2: Glitch caused by Hazard

adders, amplify this problem. In typical combinational circuits glitching accounts for between 10% and 40% of the dynamic power consumption [43]. Hazards and therefore glitches can be avoided at the cost of more circuitry. If leakage power is a deciding factor than this might not be possible.

2.2 Design Guidelines

From the analysis of the power dissipation in CMOS circuits we can derive a few simple rules guidelines for designing ultra-low power circuits.

1. The number of output transitions has to be minimal.
2. The circuit size should be minimized.
3. Glitches cause unnecessary transitions and therefore should be avoided.

The guidelines 1 and 3 are directly influencing the dynamic power consumption and the 2^{nd} guideline is influencing the leakage power consumption. Unfortunately not all of these rules can be applied at the same time. Therefore, a compromise must be achieved. Equation 2.1 demonstrates that the dynamic power consumption P_{Dynamic} depends on the operating frequency and the number of transitions. The leakage power P_{Leakage} depends on the circuit size. In most ultra-low power applications the clock frequency is low and ranges from 500 kHz [5] to about 4 MHz [108]. At these frequencies the $\mathrm{P}_{\mathrm{Leakage}}$ becomes the dominant part of the overall power consumption and therefore minimizing the circuit size is of utmost importance. The size of the circuit is influenced by the size of the cryptographic parameters, the complexity of the arithmetic units, and by the number of flip-flops needed to store variable parameters e.g.. round keys and partial products.

The energy a given circuit consumes is $E = P \cdot t$, where t is the the time it takes for one cryptographic operation to complete. Energy can also be written as $E = (\mathrm{P}_{\mathrm{Dynamic}} + \mathrm{P}_{\mathrm{Leakage}}) \cdot t$. For a given circuit $\mathrm{P}_{\mathrm{Dynamic}} \cdot t$ is independent of the clock frequency therefore, the energy this circuit consumes at different frequencies is only depending on $\mathrm{P}_{\mathrm{Leakage}} \cdot t$. Hence, the faster a circuit completes a task, the less energy it consumes due to leakage power and its energy consumption due to dynamic power can be viewed as constant.

2.3 Power Optimization at the Technological Level

This section gives a very brief overview of the most popular power optimizations methods on the technological level. The research presented in this dissertation concentrates on power savings at the logic and architectural level as well as on the system level. We used the TSMC 0.13μm ASIC library for our ultra-low power proof of concept implementations and therefore had no access to power optimization methods at the technological level. However, how our circuits could benefit from the techniques shown in this section is a topic for future research.

2.3.1 Voltage Scaling

Supply voltage scaling is the most adopted method to reduce power consumption because of the quadratic dependence of switching power P_{SW} on the supply voltage V_{DD} (see Equation 2.2). Historically however, *constant voltage scaling* has been used till the 0.8μm generation which yields significantly smaller power savings than supply voltage scaling. Constant voltage scaling came out of the need to keep supply voltages constant at standardized levels of first 5 V and then 3.3 V.

Supply voltage scaling, also called constant electric field scaling, requires the scaling of geometric features and silicon doping levels to maintain a constant field across the gate oxide of the MOS transistor. This technology results in the lowest energy delay product if we ignore the leakage power. Reducing the supply voltage V_{DD} lowers the speed of the transistors as it depends on difference between supply voltage and threshold voltage $(V_{DD} - V_T)$. Hence, V_T has to be scaled along with V_{DD}. However, lowering V_T increases the subthreshold leakage current and is limited by noise margins. Therefore, [30] identifies the leakage power as a barrier to further scaling of V_T and supply voltage scaling in general. Another approach in shown in [23] where the circuit is operated much below the optimal supply voltage and the reduced speed is compensated for by increasing area, i.e. exploiting parallelism.

2.3.2 Dual V_t

Dual V_T ASIC libraries contain two versions of most gates, one were the transistors have a high threshold voltage and one with low threshold voltage. The supply voltage is constant. Transistors with low threshold voltage V_T are faster due to the larger difference $V_{DD} - V_T$ but their leakage power is higher. Transistors with a high threshold voltage are slower, but their leakage power is lower. This gives the designer the flexibility to use gates with low threshold transistors for speed critical paths and high threshold transistors for non critical paths. Unfortunately, we did not have access to such a library for our research.

2.3.3 Transistor Sizing

Transistor sizing is the operation of varying the size of the channel of a transistor in a combinational gate. Enlarging the channel of transistors which drive large loads increases its drive capability which in turn reduces the output transition times. The short circuit power of the driven gate is directly proportional to to the input rise and fall time τ (see Equation 2.3). This can be exploited to minimize power consumption [15]. Rich ASIC libraries provide gates with a wide variety of transistor sizes which helps the synthesis tools to obtain optimum sizes for a wide range of gate loads and therefore reduce the overall power consumption.

2.4 Power Optimization at the Architectural Level

The power optimization techniques at the architectural level were available to us and we used them extensively for our proof of concept designs. Some of the methods described in this section are available within Synopsys Design Compiler and Synopsys Power Compiler, others we applied in the hardware description.

2.4.1 Path Equalization

Path Equalization, also called Path Balancing, aims to reduce glitching in a circuit by ensuring that the delay paths to all inputs of each gate are roughly balanced. This leads to nearly aligned transitions at the inputs thus eliminating hazards which in turn reduces switching activity and therefore dynamic power consumption. Path equalization can be achieved through insertion of delay elements and local transformations.

2.4.2 Clock Gating

Figure 2.3 shows a typical implementation of a synchronous register with enable. We assume that a register is multiple bits wide and consists of one flip-flop per bit. The register is disabled when the enable signal is at logic 0. Its output is fed back to its input through the multiplexer. When the enable signal is at logic 1 the register can load new values from data in. In this design each flip-flop of the register requires a

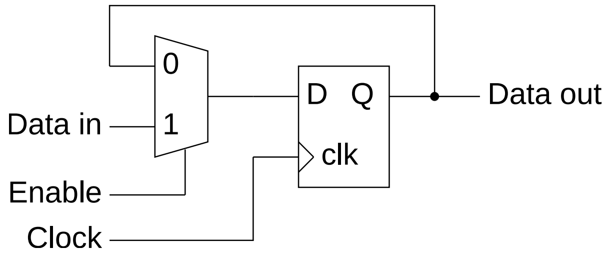

Figure 2.3: Enable Register with Multiplexer

multiplexer at its data input. Furthermore the clock network has to drive each flip-flop. Clock gating provides a way to disable the clock signals for a register, therefore eliminating the need for separate multiplexers for each input bit. Figure 2.4 shows such a design. The enable signal is usually the output of some combinatorial logic and

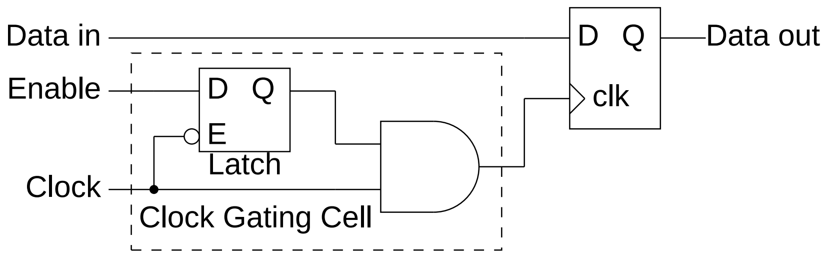

Figure 2.4: Clock Gated Register

my contain glitches. The latch prevents glitches from the enable signal to propagate to the clock input of the register. The AND gate performs the actual gating.

Clock gating replaces the multiplexers with a single clock gating cell and isolates the register clock from the global clock. The clock gating cell, containing a latch and an AND gate, consumes more power than a single bit multiplexer. However, when this technique is applied to multiple bit registers it can conserve both, static and dynamic power. We observed savings even at registers that were only 8-bits wide.

2.4.3 Operand Isolation

Operand Isolation is a method to selectively stop data from entering a block of complex combinatorial logic, causing many transitions and therefore dynamic power consumption, when the output is discarded by either an unselected multiplexer or a currently disabled register. Figure 2.5 shows an example where changes to the input A consume power even when the output A' is not used. To prevent this unneces-

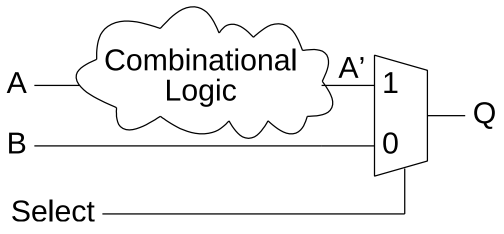

Figure 2.5: Design without Operand Isolation

sary power consumption isolation logic can be added at the input to the complex combinatorial logic. It prevents changes to input A from propagating through the combinatorial logic. The isolation logic usually consists of either AND or OR gates depending on the specific application. The example in Figure 2.6 uses an AND gate for operand isolation. The combinatorial logic only receives the input A when its output A' is selected by the multiplexer. Otherwise its input it 0.

The ideal candidates for operand isolation are complex circuits like adders, multipliers, etc., where the additional leakage power caused by the isolation logic is smaller than the dynamic power consumption of a not isolated circuit. Another good application are circuit parts that have to be active only once after many clock cycles.

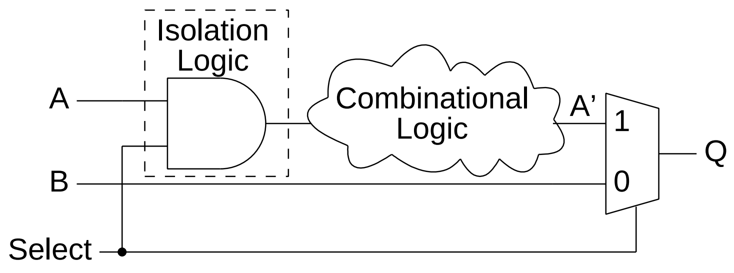

Figure 2.6: Design with Operand Isolation

2.4.4 Re-timing

Retiming for low-power is the process of positioning new or moving existing flip-flops so that they separate parts of the circuit that cause glitching from parts that have high input capacitance. As glitches do not get propagated through flip-flops this technique significantly reduces the switching activity of the high input capacitance part of the circuit and hence reduces the dynamic power consumption.

2.4.5 Local Transformations

Many of the local transformation techniques replace a gate, or a small group of gates in order to reduce dynamic power consumption from nets with higher switching capacitance. These techniques are being executed directly by the logic synthesis tools, like Synopsys Power Compiler [138]. Here is a brief overview.

Re-factoring Combinational logic functions that can be expressed through logic equations can be factored using Boolean algebra resulting in optimized expressions. In order to maximize efficiency, it is important to look across boundaries in a hierarchical design to find logic equations that can be optimized. For example, the expression $p = a \cdot c + a \cdot d$ is computed in one design block, $q = b \cdot c + b \cdot d$ is computed in another design block. The top level design, which instantiates both design blocks, computes $r = p + q$. If we look across the hierarchical boundaries we can simplify this expression to $r = (a + b) \cdot (c + d)$ which requires less gates and therefore consumes less leakage and dynamic power.

Re-mapping tries to remove high activity nets by mapping the logic function accomplished by simple logic gates into more complex gates. Figure 2.7 shows an example where the high activity net n is removed by mapping onto an OR-NAND gate.

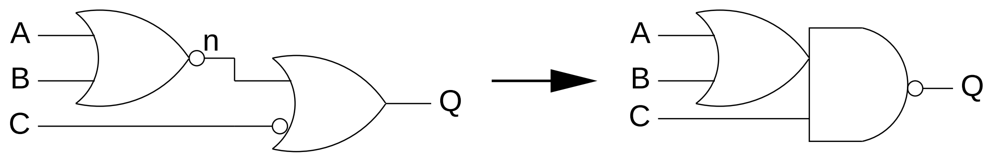

Figure 2.7: Re-mapping

Phase Assignment is used to eliminate high activity nets by changing the phase of the inputs and outputs. We say that a port, output or input, is in positive phase when no inverter appears on that port. We say that it is in negative phase if an inverter is connected. Note, this does not change the function of the circuit or lead to the output being inverted. Figure 2.8 shows how this change in phase assignment eliminates the high activity net $n2$.

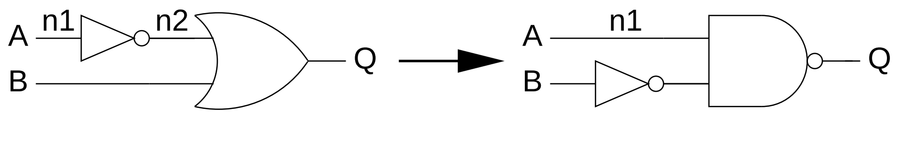

Figure 2.8: Phase Assignment

Pin Swapping Gates with multiple functionally equivalent inputs, e.g. 4 input NAND gates, might have different input capacitances associated with each input. Pin Swapping swaps the inputs in such a way as to map high activity nets to inputs with low capacitances and low activity nets to inputs with higher capacitances.

Re-sizing Resizing can be applied at the transistor and at the gate level. Here we are only considering gate resizing. Many ASIC libraries have several versions of the same gate, from slow, with higher input capacitances slowing down signal propagation and consuming less leakage power, to fast, with lower input capacitances speeding up signal propagation and consuming more leakage power. Gate resizing exploits these features to reduce the dynamic power consumption of a circuit. It does this by using fast gates on delay critical paths and slow gates on non critical paths [44]. This technique leads to more balanced paths, similar in sense to the path equalization technique discussed in Section 2.4.1, which results in a circuit with less spurious transitions, hence lowering the dynamic power consumption. However, as resizing does not change the topology of the circuit, it is guaranteed to be as fast as the original one.

2.4.6 Serialization

Serialization is a powerful technique for implementing algorithms provided they are serializable. It allows the implementer to scale the designs from high speed full parallel computing with full operand length to word serial and bit serial one bit at a time. This technique is applicable to many arithmetic circuits, e.g. multipliers, and algorithms that are of iterative nature.

Table 2.1 shows a design space exploration which, at a very high level, compares full parallel, word serial, and bit serial designs in terms or power consumption, speed, and energy consumption. Speed refers to the number of clock cycles needed for completion, the clock frequency is assumed to be constant.

The bit serial approach minimizes the number of gates which reduces the number of gate inputs and therefore input capacitances and reduces wire lengths leading to the least amount of dynamic power consumption. The small number of gates also leads to the least amount of leakage power. However, it needs the most amount of

Table 2.1: Power–Energy Tradeoff with Serialization (f-constant)

Design	Power Dynamic	Power Leakage	Speed	Energy P_{Dyn} Dominant	Energy P_{Leak} Dominant
full parallel	high	high	high	medium	low
word serial	medium	medium	medium	medium	medium
bit serial	low	low	low	medium	high

clock cycles, leading to the longest computation time. Full parallel circuits, on the other hand, have the highest power consumption but are the fastest.

If we assume that the clock frequency is constant for all designs, then the energy consumption is independent of the level of serialization if the dynamic power consumption is dominant. If the leakage power is dominant, then the energy consumption is inverse proportional to the power consumption.

2.4.7 Precomputation

This method selectively precomputes the output logic values of a circuit one clock cycle before they are required. Using these precomputed values reduces switching activity in the succeeding clock cycle. This by itself does not reduce the overall dynamic power consumption of the circuit as the total switching activity remains about the same.

We can take advantage of precomputation if it is possible to precompute a result using only a subset of input conditions. If the output could correctly be precomputed, the the original circuit can be "turned off" in the next clock cycle and if only for those cases were the precomputation yield a correct result the circuit remains "on". By carefully selecting the precomputation logic, i.e. selecting the subset of input conditions for which the output should be precomputed, the dynamic power consumption can be reduced at the cost of some additional gates with their associated leakage power consumption. This method is described in detail with intuitive examples in [4]. It also shows how to use precomputation in combinational circuits.

In arithmetic circuits precomputation can precompute values that are repeatedly used by the algorithm and store them in registers. During the actual computation the registers are use like a lookup table, which speeds up the calculations and decreases the dynamic power consumption at the cost of more registers and therefore an increased consumption of leakage power. The square-and-multiply algorithm [134] for computing the x^b is a good example where precomputation yields power savings (also called k-ary method). Assume b is an n-bit integer. The square and multiply algorithm uses on average $n - 1$ squarings and $\lceil \frac{1}{2}n - 1 \rceil$ multiplications. If we precompute 2^k values, i.e. we compute $x^0, x^1, \ldots, x^{2^k-1}$, and use them as a lookup table we can reduce the number of multiplications.

2.5 Design Flow for Ultra-Low Power

We described all of our proof of concept designs in VHDL or Verilog using the architectural and logical level optimizations described above extensively. Our target was the TSMC 0.13 μm ASIC library, which is characterized for power. We used the Synopsys tools Design Compiler [136] and Power Compiler [138] for synthesizing our designs and Modelsim for simulation. The design flow is shown in Figure 2.9. At each level of design flow we refined our architectures by analyzing the simulation results.

The first level is called register transfer level, or RTL for short. Synopsys creates a forward-annotated switching activity interchange format (SAIF) file containing directives that determine which design elements to trace during simulation. Modelsim reads this file and tracks the switching activity while simulating our design with instructions from the testbench. The results are written into a back-annotated SAIF file. Synopsys reads this file and computes first estimates of the power consumption of each section of the circuit.

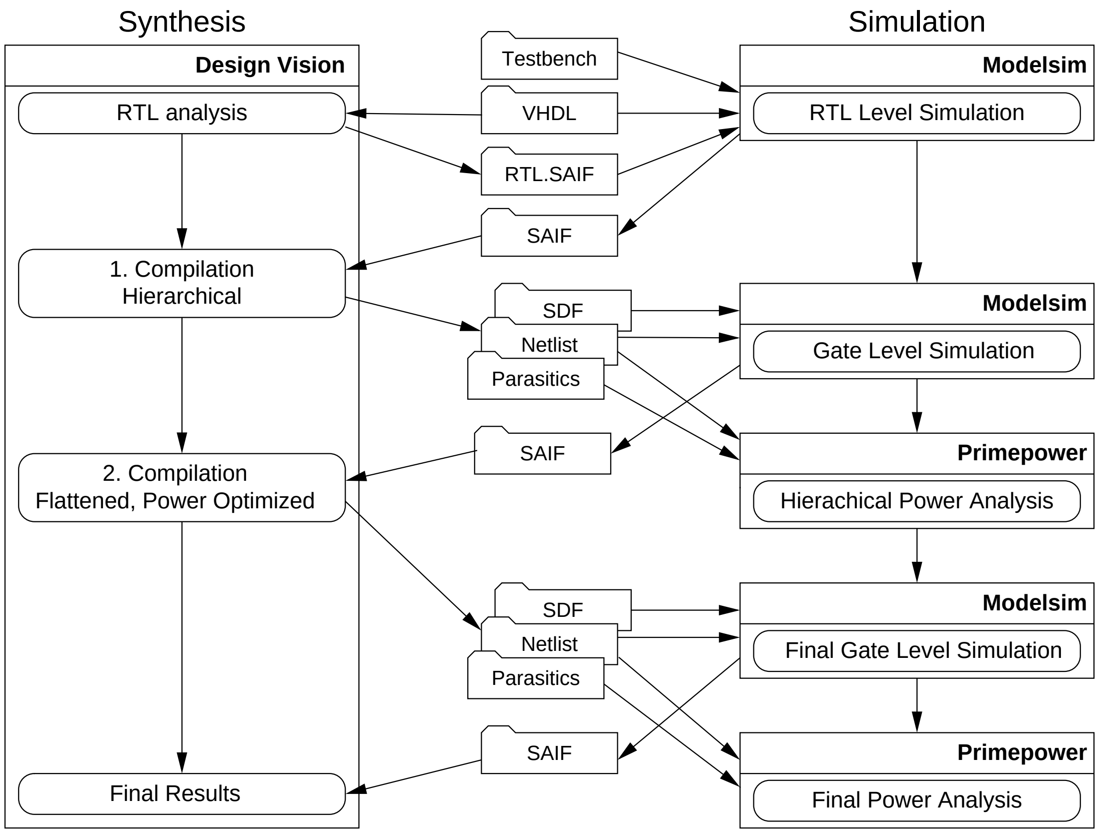

Figure 2.9: Low-Power Design Flow

The next step is to compile the netlist of the design using Synopsys which also produces the back-annotated file in standard delay format (SDF). This file contains timing information for each verilog cell in the design. Modelsim uses this for simulation and to create an accurate back-annotated SAIF file. This file is used by Primepower which provides us with detailed power consumption information for all elements of the hierarchical design.

The back-annotated SAIF file is used by Synopsys for an incremental compilation with power optimization. The result is simulated again and the resulting SAIF file is used by Synopsys for generating the final power reports. At this level we can use Primepower to show us how the power consumption varies over time.

Chapter 3

Survey of Cryptographic Algorithms

This Chapter gives an overview of cryptographic primitives present in classic and emerging algorithms and analyzes their suitability for ultra-low power implementations. This analysis is supported with data obtained from our own example implementations of ultra-low power primitives of block ciphers (Chapter 6), hash functions (Chapter 4) and even public key algorithms (Chapter 5). To this end, we introduce useful techniques for implementors of cryptographic algorithms as well as guidelines for cryptographers to design new cryptographic algorithms specifically for ultra-low power applications.

3.1 Survey

As a first step we examine current cryptographic algorithms. We limit ourselves to current popular algorithms and also include some that we found to be particularly interesting for this purpose.

3.1.1 Block Ciphers

We start our discussion by identifying common functions and structures that are shared by a wide variety of established block cipher designs. Our list of algorithms consists of classic block ciphers (DES/3DES, IDEA, RC5), the AES finalists, and the extended tiny encryption algorithm XTEA. Table 3.1 contains a summary for the ciphers under consideration.

Table 3.1: Common Elements in Block Ciphers

Algorithm	Reference	No. of Rounds	Block Size (bits)	Feistel	SP-Network	Arithmetic S-Box	Pseudorandom S-Box	Fixed Permutations	Addition (mod 2^w)	Fixed Shift/Rotation	Var. Shift/Rotation	Mod. Multiplication	Const. Multiplication
DES/3DES	[96]	16/48	64	●			●	●		●			
IDEA	[76]	8	64	●					●			●	
RC5	[122]	12/16	32/128	●					●		●		
AES	[97]	10/12/14	128		●	●		●		●			●
RC6	[120]	20	128	●					●		●	●	
MARS	[18]	2×16	128	●			●		●	●	●	●	
Serpent	[6]	32	128		●		●	●		●			
Twofish	[128]	16	128	●			●		●	●		●	●
XTEA	[152]	≥ 32	32	●					●	●			

Round Structure Virtually all modern block ciphers are iterated product ciphers, i.e. the encryption process consists of repeated application of a round function. The round function is composed of multiple layers of transformations that perform substitution and permutation, more generally also called confusion and diffusion layers. In and by itself the round function is not considered secure, but each additional round adds to the security level.

Popular round structures are variations of the Feistel network (DES, RC5, MARS, etc.) in which the round function typically only modifies part of the round data. Therefore, some publications refer to rounds in Feistel ciphers as half-rounds. Substitution and Permutation Networks (SPN) on the other hand typically modify the entire dataset in each round, examples are IDEA, Rijndael (AES), etc.

Substitution Functions All product ciphers employ substitution functions, so called S-Boxes, in one form or another for introducing non-linearity into the encryption process. Various techniques range from look-up table based pseudo-random substitutions to non-linear arithmetic functions of high degree.

Permutation Product ciphers combine various transformations for confusion and diffusion. The latter is achieved either through fixed permutations (e.g. IP/IP^{-1} and P-Box in DES) or through data dependent (variable) shifts and rotations (RC5/6 and MARS).

Key Mixing Almost all block ciphers add subkeys into the round data using XOR operations, which are fast and introduce virtually no overhead in both software and hardware implementations. Several algorithms also use a blend of XORs and regular integer addition. This has the effect that the resulting addition is no longer commutative, thereby complicating cryptanalysis.

Arithmetic Operations Certain arithmetic operations are useful for combining diffusion and non-linear mixing of round data with key bits. A good example is truncated integer multiplication which is used in MARS.

3.1.2 Stream Ciphers

Stream ciphers can be built in two different ways. The predominant method is to use a pseudo random number generator (PRNG) to generate a key stream and XOR its output with the datastream. The other possibility is to use a dedicated stream cipher like RC4.

The RC4 is a proprietary stream cipher. It uses integer addition modulo 256 and employs a dynamic S-Box with 256 8-bit entries. This S-Box is a lookup table where the entries change with each encrypted byte depending on the key.

PRNGs can be built from block ciphers, hash functions, modular exponentiators, or linear feedback shift register (LFSR) based stop-and-go generators. The first three have a big advantage over a dedicated stream cipher, namely, the base function can also be used for its original purpose. The stream cipher functionality comes at minimal extra cost.

3.1.3 Hash Functions

A hash function produces a short fixed size digest of a long message. This digest can be used to check the integrity of a message. Hash functions that employ a secret key are called message authentication codes (MAC) and hence provide message authentication as well as integrity. Universal hash function families provide provable security and can be used to build provable secure MACs [148], i.e. bounds on the success probability of an attacker independent of the computational power applied can be proven. For this article we selected some of the most popular hash functions and additionally two universal hash function families: NH [14] and WH (Chapter 4). Table 3.2 summarizes some hash function parameters.

Each hash function has a fixed input size. Longer input strings are split, shorter ones are padded. The hash size specifies the length of the resulting hash value, independent of the length of the input string. The hash functions MD4, MD5, and those described in the Secure Hash Standard (SHS) SHA-1, SHA-256, etc. all belong to the same group of hash algorithms. MD2, MD4, MD5, and SHA-1 share similar functional blocks with minor differences in the parameters. In spite of recent advances in attacking SHA-1, it is still in wide spread use, however MD4 and MD5 are considered compromised.

NH was introduced as a new hash function family for an authentication code called UMAC and is based on modular integer multiplication and summation. WH is a hash function family with even stronger security properties than NH and can be used in place of NH. It is specifically designed for implementation in ultra-low power hardware and is based on modular polynomial multiplication and summation. NH and WH are defined for any fixed block size. For our table we assume a block size of 64 bits as this was also the size chosen in [153] for the implementation.

Table 3.2: Summary of Hash Function Characteristics

Algorithm	Reference	No. of Rounds	Input Size(bits)	Hash Size(bits)	Constants (bytes)	Variables (bytes)	Integer Multiplication (bits)	Integer Addition (bits)	Polynomial Multiplication	Simple Logic Functions	Fixed Shift/Rotation
MD2	[66]	16	128	128	256	48		32		●	●
SHA-1	[99]	4	512	160	16	20		32		●	●
MD4	[118]	3	512	128	16	16		32		●	●
MD5	[119]	4	512	128	16	16		32		●	●
NH	[14]	n/a	64	64			64	64			
WH	[153]	n/a	64	64					●	●	

3.1.4 Public Key Cryptosystems

The security of classic public key algorithms typically relies on the hard problem of integer factorization or finding the discrete logarithm (DL) in a finite field (RSA, ElGamal, etc.). The dominating arithmetic operation of these algorithms is modular exponentiation, which is typically implemented using modular multiplication and squaring operations. A variant of RSA is Rabin's Scheme, in which the public key exponent is fixed to the value 2. This significantly reduces the complexity of the encryption operation to modular squaring and allows for a compact implementation, whereas the general case of RSA would be too large. The security of Rabin's Scheme relies on the integer factorization problem just like RSA. Therefore, there is no compromise in terms of security.

Elliptic curve cryptography (ECC) uses a variant of the DL problem that is defined over the additive group of points on an elliptic curve. Points are repeatedly added (or doubled) until a scalar 'multiple' of the originating point is obtained. A single point "addition" consists of a heterogeneous variety of finite field operations such as addition/subtraction, multiplication and in some cases inversion.

Hyperelliptic curve cryptography (HECC) has been proposed as a generalization of ECC. Its advantage is that operands can be even shorter than for ECC while an equivalent level of security is maintained. The problem of HECC is its arithmetic structure which is even more complex than that of ECC. While efficient explicit expressions exist for the group operations [106], they still contain diverse arithmetic primitives that may prove to be too complex for ultra-low power implementations.

NtruEncrypt is based on a completely different hard problem namely the Shortest Vector Problem (SVP) in high dimension lattices. The central arithmetic primitive of NtruEncrypt is multiplication in a truncated polynomial ring. The operation itself can be subdivided into individual computations of polynomial coefficients through accumulation of partial products. Table 3.3 compares the algorithm parameters of

Rabin's Scheme, Ntru and Elliptic Curve of equivalent security levels.

Table 3.3: Comparison of PKC Functions

Algorithm	Reference	Encryption	Signature	Message Payload (bits)	Ciphertext (bits)	Signature Length (bits)	Integer Multiplication (bits)	EC Point Addition (bits)	Polynomial Coefficients (bits)
Rabin's Scheme	[121]	●	●	< 512	512	512	512		
NtruEncrypt	[59]	●		< 265	1,169				8
NtruSign	[58]		●			1,169			8
EC-MV	[134]	●		< 200	400			169	
EC-DSA	[64]		●			200		169	

3.2 Analysis

In this section we analyze how the structure, functional primitives and storage requirements of cryptographic algorithms relate to their energy consumption. From this we can devise recommendations for future algorithms tailored toward ultra-low power implementations. For our example implementations, we used the TSMC 0.13μm ASIC library, which is characterized for power, and the Synopsys tools Design Compiler and Power Compiler for synthesis. Modelsim was used for simulation and capturing of switching activity which was used by Power Compiler for power estimation. We noticed that at a clock frequency of 500 kHz, which is commonly found in sensor nodes, the static power consumption P_{Leak}, caused by leakage, outweighs the dynamic power consumption P_{Dyn}, caused by switching activity. Future CMOS generations will have even higher leakage power consumption according to the International Technology Roadmap for Semiconductors [63] which will cause it to be dominant even at higher frequencies.

3.2.1 Algorithm Structure

The algorithm structure can tell us how well each algorithm lends itself to both parallelization and serialization. The latter ties in directly with minimization of circuit area and therefore static power consumption.

Scalability refers to the possibility of scaling an algorithm between bit serial and highly parallelized realizations in an efficient manner. In certain contexts, such as Public Key Cryptography, scalability may also encompass the re-use of existing processing elements for higher precision operands than originally intended, i.e. using a 1024 bit modular exponentiator hardware for 2048 bit operands.

The iterative round structure of most block ciphers is by itself an indicator for a reasonable degree of scalability, provided that all rounds are the same. Then only one instance of the round function needs to be implemented. Serialization is possible with RC5 and RC6 and, ignoring a slightly modified last round, also for most other ciphers.

In contrast to block ciphers, public key schemes are for the most part based on arithmetic over large integer or polynomial fields, which usually lends itself well to serialization, even though certain operations like modular reduction introduce additional complexity which hinders serialization beyond a certain point. The biggest problem with serial implementations is the running time that is cubic in the operand size for most public key algorithms. Insofar it is important to evaluate the trade-off between area usage and a certain degree of parallelization.

Modularity is closely related to the concept of scalability, but more in the sense that simple processing elements can be replicated easily for further parallelization of a task when higher performance is necessary. We will give our implementation of the NtruEncrypt algorithm as an example (Chapter 4). NtruEncrypt has good modularity. Its basic operation can be subdivided into computations using 8-bit long polynomial coefficients. By intelligently arranging memory accesses to these coefficients, it is possible to perform computation of multiple coefficients of the result in parallel. Since storage of operands is by far the largest portion of the circuit, scaling up the number of parallel arithmetic units (AUs) has little effect ($< 50\%$ increase) on the overall area and power consumption. At the same time the number of clock cycles can be reduced dramatically, as can be seen from Table 3.4. The critical path delay is not affected.

Table 3.4: Serial / Parallel Tradeoffs in NtruEncrypt (500kHz, $N = 167$)

Implementation	**Power (μW)**			**Area**[1]	**Clock**	**PDP**
	$\mathbf{P_{Dyn}}$	$\mathbf{P_{Leak}}$	**Total**		**Cycles**[2]	**ns$\times\mu$W**
Single AU	4.03	15.1	19.1	2,850	29,225	13.18
8 AUs in parallel	5.00	22.5	27.5	3,950	3,682	18.96

Regularity describes the degree of similarity between modules at different levels of parallelization. At the logic level highly regular designs allow for efficient parameterization and reuse, while irregular circuits often require manual design changes. At the algorithmic level a high degree of regularity expresses the uniformity of operations necessary to perform a task, while very complex tasks consisting of many different atomic operations are characterized by their irregularity. Rabin's Scheme, NH and WH are good examples for algorithms that have high regularity. They all have one simple underlying function. In block ciphers even a single round of a block cipher

[1] Area is given in terms of equivalent two input NAND gates

[2] Number of clock cycles to complete one operation

can take up considerable amounts of chip area. Serialization of the round function is therefore necessary. Ciphers with a homogeneous round function (e.g. AES) have a high degree of regularity and therefore seem better suited for serialization than others of heterogeneous structure (e.g. DES).

Power - Energy Tradeoff Energy equals the amount of power dissipated over time. Increasing the degree of parallelism increases the power consumption, but at the same time decreases the computation time. The trade-off depends on the overall structure of the architecture, since certain elements may have constant size. The point of optimality can be found by modeling the energy consumption as a function of the degree of parallelism. A useful metric for this is Energy per Bit Encrypted. It describes the amount of energy necessary to encrypt a single bit of the message. This metric can be used to compare the energy efficiency of cryptosystems at an equivalent level of security. It is independent of the actual operand length.

3.2.2 Functional Primitives

In the following we describe the characteristics of each group of primitives we encountered in Section 3.1 and their suitability for ultra-low power implementation.

Simple Logic Functions This group contains logic functions where the output is dependent on only a small, fixed set of inputs. This includes functions like XORs of two bitstrings (AddRoundKey in AES), bit-multipliers and multiplexers. The number of logic gates scales linearly with the width of the data path.

Fixed Shifts and Permutations Fixed in this context means that the shifts and permutations are not data dependent. Permutations and expansions are used in block ciphers (DES, Serpent, AES) as non-linear diffusion elements. Fixed shifts and rotations serve the same purpose and are frequently used in block ciphers (AES, Mars, Serpent, Twofish) and hash functions (MD2, SHA-1, MD4, MD5). Common to all these functions is that their implementation introduces virtually no cost as they require only wiring resources and no logic. They are perfect for any hardware implementation.

Data Dependent Shifts Also called "variable shifts" for short, are used in a couple of block ciphers (RC5, RC6, Mars) due to their resistance against differential cryptanalysis. Implementations frequently use barrel shifters to support all possible shifts or rotations. The delay for one shift operation is proportional to $\log_2 n$ but its area scales with $n \log_2 n$. For situations in which the shift or rotation is followed by a register, it may be more power efficient to instead implement the register as a shift register with parallel load and combine it with some additional control logic and a counter. Due to the relatively high area cost variable shifts are not well suited for ultra-low power implementations unless they can be combined with existing registers.

Integer Arithmetic Integer arithmetic primitives such as addition and multiplication are frequently the most costly functions in a cryptographic algorithm. Although they can often be implemented in a bit serial fashion (e.g. multiplication), the efficient propagation of carries presents itself as a major problem. The simplest form of an adder, the carry propagate or ripple carry adder, scales linearly with the word size n, but glitches in the carry chain cause high dynamic power consumption. Various alternatives exist in the literature, but they go along with a penalty in terms of area and therefore static power consumption. Because of these costs integer arithmetic should be avoided as much as possible for new ultra-low power algorithms. An elegant solution to this problem might also be the usage of residue number systems which can result in carry free arithmetic.

Arithmetic primitives are frequently combined with modular reduction steps. In trivial cases the modulus is chosen as 2^k, which means that only k bits of the result are kept, excess bits are truncated. Finite field arithmetic with a non-trivial modulus adds a fair amount of complexity to the circuit. Simple implementations perform conditional subtractions of the modulus from the result, depending on the value of its most significant bit. For certain classes of public key algorithms which heavily depend on modular arithmetic, the use of residue number system arithmetic has proven to be effective for efficient implementation.

Polynomial Arithmetic Polynomial arithmetic, i.e. arithmetic in extension fields, is preferable for ultra-low power implementation due to limited carry propagation and improved regularity. Therefore, several algorithms are specifically tailored towards arithmetic in $GF(2^k)$ (e.g. AES). Additions in fields of characteristic two can simply be implemented by an XOR. For ultra-low power applications multiplication may be implemented in a bit-serial fashion. This is facilitated by the simplicity of the addition and reduction steps.

The vast difference in power consumption between integer and polynomial arithmetic is demonstrated in Chapter 4, where we describe the universal hash function families NH and PH. PH is a redefinition of NH which uses polynomials over $GF(2)$ instead of integers. We would like to emphasize that both hash functions are 2^{-w}-almost universal, hence both provide the same level of security. The differences in area, speed and power consumption, however, are impressive. Note, both algorithms require only 64 clock cycles for computing the hash of one set of inputs. The results of our implementation at 500 kHz are summarized in Table 3.5.

Table 3.5: Comparison of NH (integer) and PH (polynomial) Implementations

Implementation	**Power (μW)**			**Area**[3]	**Delay**	**PDP**
	$\mathbf{P_{Dyn}}$	$\mathbf{P_{Leak}}$	**Total**		**ns**	**ns$\times\mu$W**
NH (integer)	5.47	28.1	33.6	5,291	9.92	333.31
PH (polynomial)	3.41	12.1	15.5	2,356	1.35	20.93

Substitution Functions (S-Box) can be implemented using various techniques. While look-up tables are fast and easy to implement, the size of the tables is often prohibitively expensive. If performance is secondary to low power consumption and an arithmetic description for the S-Box exists, then a circuit realization of the underlying arithmetic operation may be preferable. We performed a case study on different AES S-box architectures for ultra-low power implementations. For one circuit we implemented the S-box as an arithmetic function using its inherent algebraic structure. For our other circuit we implemented a 256×8-bit look-up table in combinational logic. The results are summarized in Table 3.6. Even though the combinational implementation uses only 30% of the dynamic power of the arithmetic implementation, due to its size its total power consumption is two times higher. This shows that it is advantageous if the content of the S-box can be described algebraically, which additionally gives an opportunity for serialization.

Table 3.6: Comparison of AES S-Box Implementations

Implementation	**Power (μW)**			**Area**[4]	**Delay**	**PDP**
	$\mathbf{P_{Dyn}}$	$\mathbf{P_{Leak}}$	**Total**		**ns**	**ns$\times\mu$W**
AES S-Box: Logic	0.42	7.67	8.10	1,397	1.61	13.04
AES S-Box: Algebraic	1.39	2.68	4.07	431	4.68	19.05

3.2.3 Storage Requirements

Storage requirements of cryptographic algorithms are manifold. All constants and variables used by an algorithm as well as implementation specific storage elements add to it. Constants are comprised of fixed setup parameters, precomputed constants, and static S-Boxes. Fixed parameters and precomputed constants can be implemented in

[3]Area is given in terms of equivalent two input NAND gates

[4]Area is given in terms of equivalent two input NAND gates

combinational logic. Strategies for larger sets of constants i.e. S-Boxes are described above in Section 3.2.2.

Variables, as well as variable S-Boxes (RC4) and temporary data has to be stored in registers or RAM. Pipelining techniques require additional storage elements. Since storage elements typically impose significant area and power penalties, they should be used conservatively in ultra-low power implementations.

3.2.4 Implementation Considerations

Here we want to mention additional considerations that go beyond looking at the structure and elementary functions of a cryptographic algorithm.

Multi-encryption and Multi-hashing Multi-encryption /-hashing are two related concepts for increasing the security of an algorithm by applying it repeatedly. The Triple Data Encryption Algorithm (TDEA) also known as Triple DES is probably the best known example of multi-encryption. It applies DES three times in a row, using either two or three different keys depending on the keying option. It was originally developed out of the need to prolong the lifetime of DES until a new, more secure standard was found. However, in the light of ultra-low power cryptography, multi-encryption and multi-hashing can be seen as an enabling technology. It makes it possible to use block ciphers or hash functions that consume very little power but have a small security margin, and run them several times in series, thus obtaining a more secure overall cipher or hash function.

Fixed or Constant Parameters in cryptosystems can help to alleviate the problem of large storage requirements, and even simplify certain computations. This is highly dependent on the intended application context. For example, an Internet server will typically have to change keys and associated key parameters frequently, such that keeping them constant is not possible. In embedded applications, where communication is typically limited to links between sensor nodes and a base station, fixing parameters such as the public key helps to reduce the storage requirements significantly.

Precomputation is a powerful method for solving latency problems and is especially important for low-power nodes where intensive computations must be spread over time to reduce the power consumption below the maximum tolerable level. If the algorithm allows precomputation of intermediate results and thus only a small number of computations are necessary for the processing of the data, then latency may be virtually eliminated.

3.3 Recommendations for Designing new Algorithms

In this section we outline recommendations for designers of cryptographic algorithms from an implementers point of view. Note that designing cryptographic algorithms requires a unique skill-set and many years of experience and therefore should be attempted only by professional cryptographers. We use the terminology defined in the previous section to describe the key features that a new algorithm should have. The goal is to obtain an ultra-low power, scalable, secure algorithm.

The most important requirement is scalability. It should be possible to scale the algorithm from a bit serial implementation to a highly parallel implementation depending on the desired maximum power consumption and speed. The extent to which an algorithm is scalable depends on its regularity. New cryptographic algorithms should be regular and contain only a limited amount of different primitives. In order to further improve the scalability, the basic functions themselves should be serializable. Then the implementor can trade off speed with power on a fine level of granularity and not just on the algorithmic level (e.g. round function).

Serializing an algorithm slows down its operation assuming the clock speed is held constant. However, in environments where data has to be sent quickly but only once in a while, it would be desirable if most steps could be computed "offline" ahead of time. When the data becomes available only a simple, fast computation should be required to complete the operation (e.g. addition of data to the key).

Applications for WSN and RFIDs have a variety of security requirements ranging from high risk applications e.g. military target tracking, where the devices are likely to being attacked to low risk applications e.g. passive environmental monitoring. This would be best supported if the new algorithms could operate with various key lengths (e.g. AES). Another method to increase the security level without increasing its footprint is multi-hashing / multi-encryption.

We would like to briefly summarize the implementation considerations for elementary functions.

- Lookup tables are costly. An algebraic representation can be more efficient.
- Polynomial arithmetic in $GF(2^k)$ is well suited for hardware implementation.
- Integer arithmetic has an inherent high power consumption.
- Data dependent shifts and rotations are costly unless they can be combined with existing registers.
- Fixed shifts and rotations are well suited for hardware implementation.

Messages in WSN and used by RFIDs are usually very small and average between 30 bits to 100 bits in length. The power consumed by transmitting a bit is high compared to the power consumed by computation. A new algorithm should therefore have a compact representation of cipher I/O. Encryption functions should not cause any message expansion and use a small block size. Hash functions should result in small digests as they are transmitted in addition to the original data. Ideally, the digest size should not affect the collision probability, i.e. security.

The challenge of future research is to find an algorithm that at its core contains a a simple, scalable primitive. It would be highly desirable if this simple primitive can be used as a common element for secret and public key functions. This would make it possible to provide both types of functions for ultra-low power applications, which in turn will enable simpler and efficient security protocols.

3.4 Conclusion

Current wireless sensor nodes and RFID tags struggle with the load of cryptographic algorithms implemented in software. The next generation nodes will be MEMS powered and therefore their power constraints will be even more severe. Future RFID tags will incorporate more functionality like writable memory and sensors. This will further decrease the amount of power available for cryptography. In this Chapter, we provide guidelines on how to implement current cryptographic algorithms in hardware to enable sufficiently strong cryptography for these devices. We use these guidelines in the following chapters for our ultra-low power implementations.

Chapter 4

Universal Hash Functions

Parts of this chapter were presented in [153] and [68].

4.1 Motivation

Protecting the integrity of data is of utmost importance for many application scenarios. For example, smart dust motes that are embedded in a bridge could monitor the stress and inform the authorities in case of emergency. Wireless sensors might monitor plant growth, moisture and PH-value on a farm. In both cases the data is not confidential but its authenticity and integrity are very important. For this purpose, efficient *Message Authentication Codes* (MACs) [131] may be preferable over digital signature schemes [32] due to their high encryption throughput and short authentication tags. A disadvantage for both digital signature schemes and traditional MACs is that they provide only computational security. This means that an attacker with sufficient computational power may break the scheme. More severely, the lack of a formal security proof makes these schemes vulnerable to possible shortcut attacks.

Universal hash functions, first introduced by Carter and Wegman [20], provide a unique solution to the aforementioned security problems. 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 [148] 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.

Black et.al. [14] describe a new, provably secure message authentication code (called UMAC), which has been designed to achieve extreme speeds in software implementations. A hash function family named NH underlies hashing in UMAC. In this paper we improve upon NH in order to make secure hash functions possible in ultra-low-power devices. We implement NH with power efficiency guidelines in mind and notice 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 take a different approach. We make incremental changes to the original algorithm which result in improved efficiency in hardware. This leads to the new hash function families named PH and PR. We then identify the main power consumers (i.e. registers, adders) and carefully remove components one by one. We formulate the resulting new algorithm (WH) mathematically. We prove that all three new hash function families are still at least as secure as the original NH.

While WH is consuming an order of magnitude less power than NH, its leakage power consumption 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 [123] 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.

4.2 Preliminaries

4.2.1 Notations

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

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 $GF(2)$ [87] of degree less than w. Note that each message block m_i and key block k_i belongs to either U_w, P_w or $GF(2^w)$. Here $GF(2^w)$ denotes the finite field of 2^w elements defined by $GF(2)[x]/(p)$, where p is an irreducible polynomial of degree w over $GF(2)$. In this setting the bits of a message or key block are associated with the coefficients of a polynomial. To illustrate, suppose $w = 6$ and $p = x^6 + x + 1$. Let us see how two messages (binary bit strings), 101101 and 100011, can be multiplied in the Galois Field of $GF(2)[x]/(p)$. 101101 and 100011 would be mapped into $x^5 + x^3 + x^2 + 1$ and $x^5 + x + 1$, respectively. Multiplication of these two polynomials yields $x^{10} + x^8 + x^7 + x^6 + 2x^5 + x^4 + 2x^3 + x^2 + x + 1$. This is equivalent to $x^{10} + x^8 + x^7 + x^6 + x^4 + x^2 + x + 1$ (since $2x^5 \equiv 0x^5 \equiv 0$ in $GF(2)$). Dividing this polynomial by p and taking the remainder, we obtain $x^5 + x^3 + x^2 + x$ (corresponding to the bit string 101110). Note that this way carries are eliminated as well. Finally the addition symbol '+' is used to denote both integer and polynomial addition (in a ring or finite field). The meaning should be obvious from the context.

4.2.2 Universal Hashing

A universal hash function, as proposed by Carter and Wegman [20], 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 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 [100]:

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 $H = h : A \to B$ is said to be **ϵ-almost universal** ($\epsilon - AU$) if for every $M, M' \in A$ where $M \neq M'$,*

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

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 $(a-b)/(b(a-1))$.

4.3 Hash Function Families

Here, we present the universal hash family NH, and then we introduce three variations to it. Each one improves upon the previous one in terms of efficiency , but diverges further from NH.

4.3.1 NH

In the past many universal and almost universal hash families were proposed [130, 49, 123, 74, 14, 36]. Black et al introduced an almost universal hash function family called NH in [14]. The definition of NH is given below.

Definition 3 *([14]) 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 *([14]) 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.

4.3.2 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.

4.3.3 NH-Polynomial with Reduction (PR)

The main motivation for this construction is the length of the authentication tag, which 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 [49], SQUARE proposed in [36]) since reductions are relatively costly to implement in software. A modulo reduction involves division and computation of the remainder. In hardware, however, reductions (especially those with fixed low-weight polynomials) can be implemented quite efficiently. The reduction becomes an integral part of the computation and involves only a simple subtraction at each step (see Section 4.4.4).

4.3.4 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 variation 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 4.4.4.

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:*

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

Due to the scaling with the 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.

4.3.5 Analysis

In this section we give three theorems 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 obtain

$$Pr\left[\sum_{i=1}^{n/2}(m_{2i-1} + k_{2i-1})(m_{2i} + k_{2i}) = \sum_{i=1}^{n/2}(m'_{2i-1} + k_{2i-1})(m'_{2i} + k_{2i})\right] \leq 2^{-w} \ . \quad (4.1)$$

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 a ring 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 (4.1). 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

$$k_1(m_2 - m'_2) = y - m_1(m_2 + k_2) + m'_1(m'_2 + k_2) \ . \tag{4.2}$$

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 (4.2) 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 (4.2) 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.*

The intuition behind this theorem and theorem4 is that when PR or WH are used as the hash function, we can mathematically prove and quantify that the adversary cannot falsify our message with a better probability than randomly selecting the right hash value from all possible hashes.

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. For brevity we denote $(m_{2i-1}+k_{2i-1})(m_{2i}+k_{2i}) = mk_{2i}$, $(m'_{2i-1}+k_{2i-1})(m'_{2i}+k_{2i}) = m'k_{2i}$ and so on. 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

$$\begin{aligned} Pr\Bigg[\sum_{i=1}^{n/2}&(m_{2i-1}+k_{2i-1})(m_{2i}+k_{2i})\left(x^{(\frac{n}{2}-i)w}\right) = \\ &\sum_{i=1}^{n/2}(m'_{2i-1}+k_{2i-1})(m'_{2i}+k_{2i})\left(x^{(\frac{n}{2}-i)w}\right) \pmod p\Bigg] = 2^{-w} \ . \end{aligned} \tag{4.3}$$

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

$$\begin{aligned} Pr_{k_{2l-1}\in GF(2^w)}\Bigg[\sum_{i=1}^{n/2}&(m_{2i-1}+k_{2i-1})(m_{2i}+k_{2i})\left(x^{(\frac{n}{2}-i)w}\right) = \\ &\sum_{i=1}^{n/2}(m'_{2i-1}+k_{2i-1})(m'_{2i}+k_{2i})\left(x^{(\frac{n}{2}-i)w}\right) \pmod p\Bigg] = 2^{-w} \end{aligned} \tag{4.4}$$

satisfied for all $1 \leq l \leq n/2$ implies (4.3). Setting y and z as

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

and

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

we rewrite the probability expression in (4.4) as

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

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

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

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

$$\begin{aligned} k_{2l-1} &= (m_{2l}-m'_{2l})^{-1}\big((x^{-(\frac{n}{2}-l)w})(y+z)- \\ &\quad m_{2l-1}(m_{2l}+k_{2l})+m'_{2l-1}(m'_{2l}+k_{2l})\big) \pmod{p}\ . \end{aligned}$$

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,

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

□

4.4 Implementations

4.4.1 NH

The algorithm for NH is described in [14]. It is given in this paper 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 simple functional block diagram shown in Figure 4.1. 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 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 multiplication results in $mout = ma \cdot mb$. The final adder accumulates all $n/2$ products.

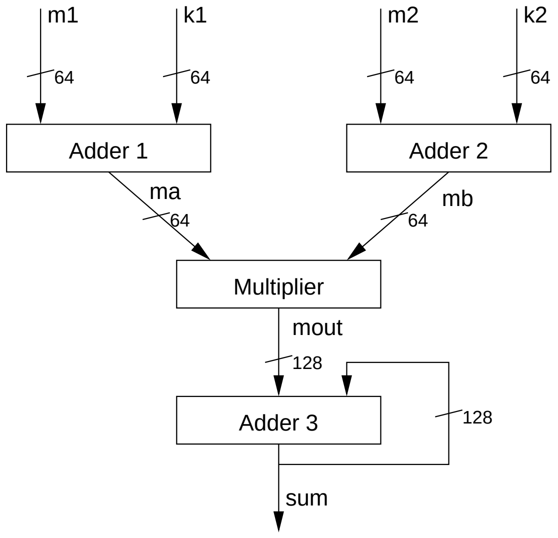

Figure 4.1: Functional diagram for NH

The detailed block diagram for the circuit is much more complex and can be found in Figure 4.2[1] As power consumption is our main concern and not speed, we base our design on a bit serial multiplier. For each multiplication of two w bit numbers, w partial products need to be computed and added: $mout = \sum_{j=1}^{w} ma \cdot mb[j] \cdot 2^{j-1}$.

This decision gives us the ability to use a bit serial adder for Adder 2 as its result $mb[j]$ (indicated as *mult* in Figure 4.2) can directly be used by the bit serial multiplier. A bit serial adder produces one bit of the result with each clock cycle, starting with the LSB, and it has minimal glitching. Unfortunately, this adder has to store both its inputs in shift registers which raises the leakage power consumption.

[1]Boxes labeled "R1" or "R2" denote shifts.

The multiplicand *ma* has to be available immediately. Therefore, we use a simple

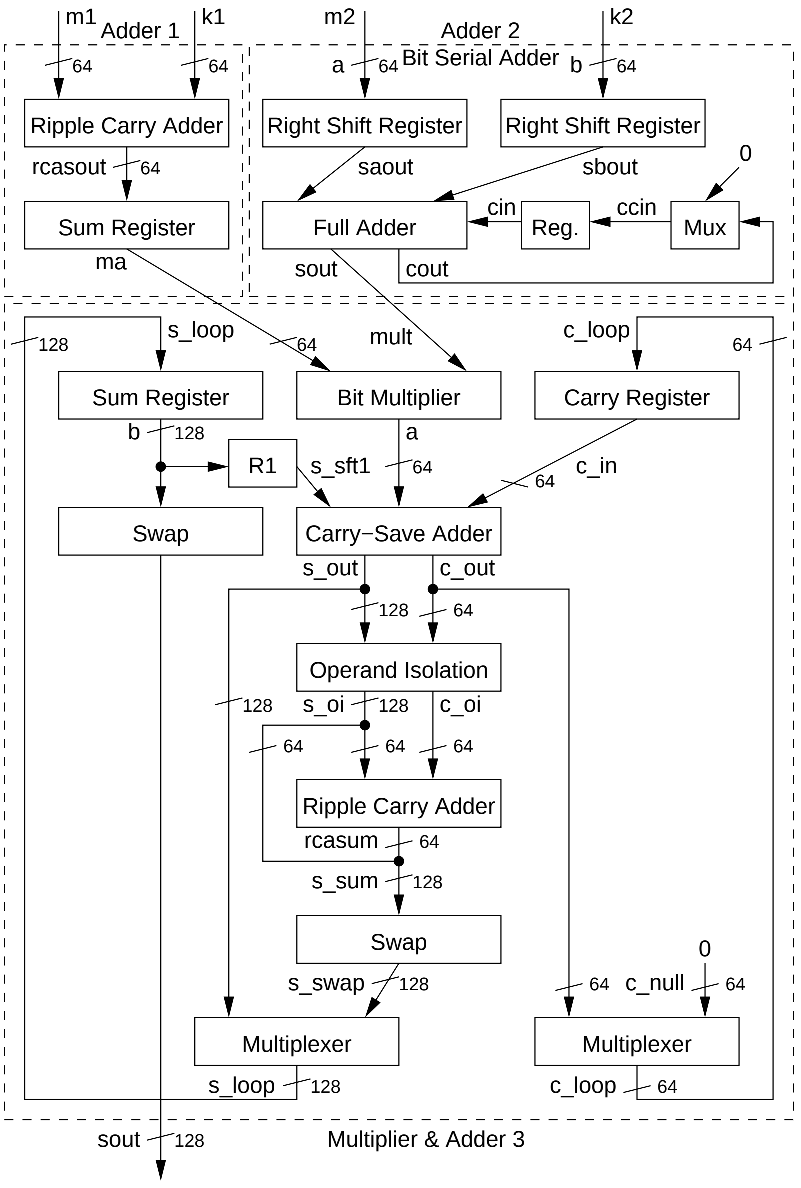

Figure 4.2: Detailed Block Diagram for NH Datapath

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. Due to its delay, it is necessary to store its output in a register. However, Adder 1 needs to compute a new result only every 64 clock cycles, hence its dynamic power consumption is tolerable.

The Bit Multiplier in Figure 4.2 computes the partial products, one during each clock cycle. The addition of the partial products is accomplished using a carry-save adder and the Right Shift Algorithm [103]. Figure 4.3 shows an example of this algorithm. This adder uses the redundant carry-save notation which results in minimal

	$1001 \cdot 1101$							
Register	0	0	0	0	0	0	0	0
4th-bit = 1		1	0	0	1			
Add:	0	1	0	0	1	0	0	0
Shift:	0	0	1	0	0	1	0	0
3rd-bit = 0		0	0	0	0			
Add:	0	0	1	0	0	1	0	0
Shift:	0	0	0	1	0	0	1	0
2nd-bit = 1		1	0	0	1			
Add:	0	1	0	1	1	0	1	0
Shift:	0	0	1	0	1	1	0	1
1st-bit = 1		1	0	0	1			
Add:	0	1	1	1	0	1	0	1

Figure 4.3: Right shift multiplication of 1001 and 1101

glitching as the carries are not fully propagated. However, this requires 64 additional flip-flops 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 4.1. 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 makes use of the Sum Register and the Carry Register of the Multiplier to store the result of Adder 3, 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 caries to the sum. This addition is done using a ripple carry adder (Figure 4.2 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.

4.4.2 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 (see Figure 4.4). Adder 1 is replaced by 64 XOR gates and its result does not need to be stored in a register because of their much shorter delay. The bit-serial adder (Adder 2) is replaced by XORs and a single 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 adder is replaced by XORs. This eliminates the whole carry path including the carry register (64 flip-flops), the multiplexer, and the ripple carry adder to realign the sum. 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.

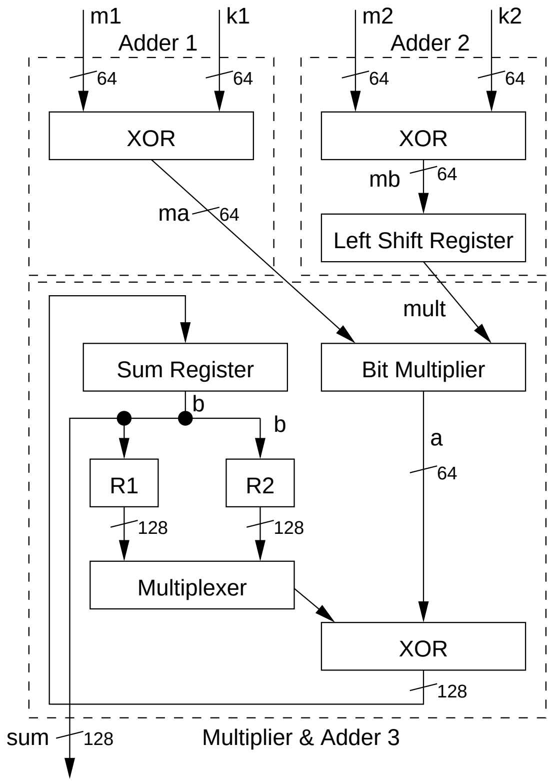

Figure 4.4: Functional Diagram for PH

4.4.3 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, eliminating the need for extra storage space for the partial product, which 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. We are performing the modulo reduction after every single addition. This keeps the reduction circuit simple and the result is never larger than w bits. However, the Multiplier and Adder 3 can no longer be merged. This is expensive as we need to have not just one adder but also one extra 64-bit register (see Figure 4.5). Therefore, we are not able to reduce the number of flip-flops in our implementation but we reduced the switching activity as our datapath through the multiplier is only 64 bits wide. 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 then that of NH.

4.4.4 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 4.1) could not be merged as in PH. However, the shorter output size of PR and hence the savings in transmission power make a modulo reduced result very appealing. Therefore, we used a different approach to optimize PR We removed from PR's implementation the 64 bit register and the XOR gates of Adder 3 and the 64 bit multiplexer from the Multiplier, The function of the resulting design is characterized by the construction shown in Definition 6:

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

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%. This dramatic savings become more obvious when the block diagrams for NH in Figure 4.2 and for WH in Figure 4.6 are compared.

4.4.5 Control Logic

The control logic manages the switching of the multiplexers, loading of the next data set and resetting of the carry registers. Due to the iterative nature of our multiplier,

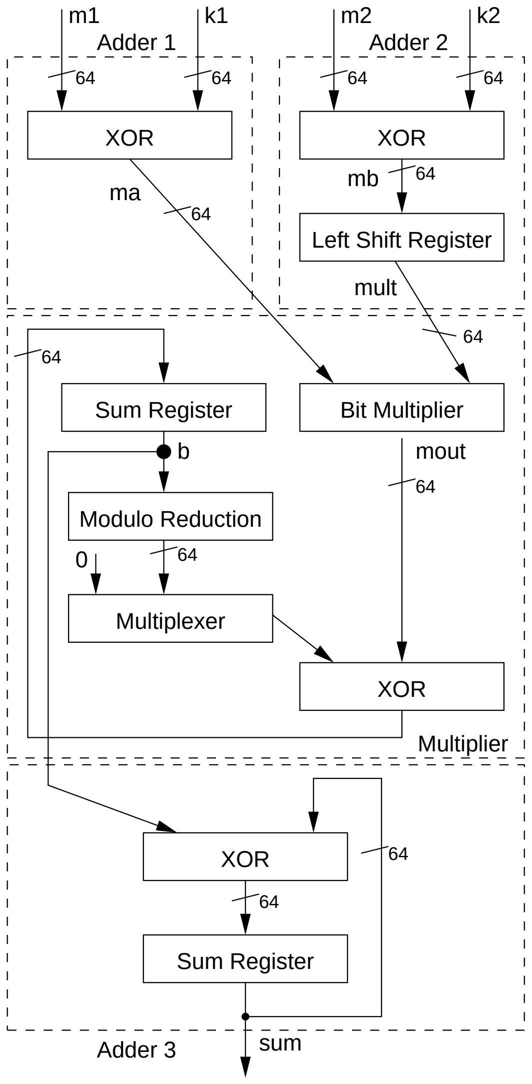

Figure 4.5: Functional Diagram for PR

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 critical. Instead of an integer counter, we use a linear feedback shift register (LFSR)

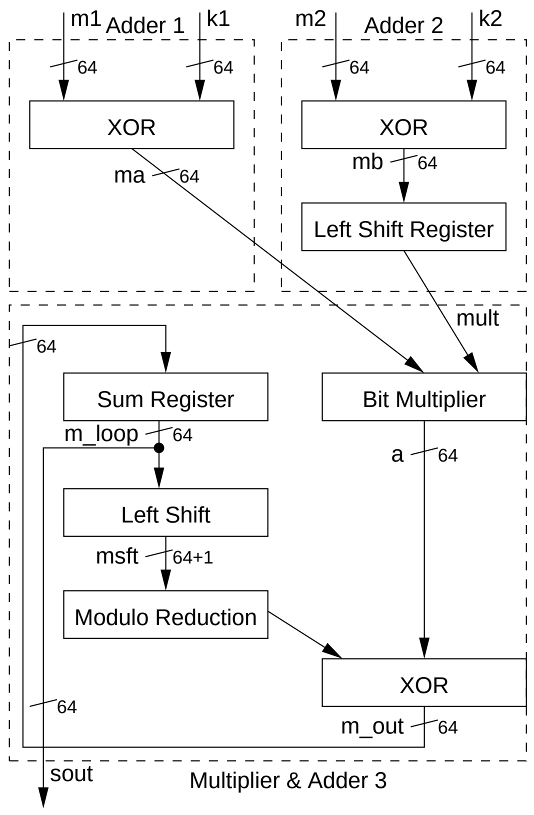

Figure 4.6: Block diagram for WH

with 6 flip-flops for NH, PH, PR, and WH-64, enhanced to "count" up to 64. LFSRs have minimal glitching and therefore make power efficient and fast counters.

4.4.6 Implementation Results

For synthesizing our designs we used the Synopsys tools Design Compiler [136] and Power Compiler [137], and the TSMC 0.13 μm ASIC library. The results of the simulation on many input sets were verified with the Maple package [53] for consistency.

Table 4.1 shows the results for power, area, and delay for the hash function implementations synthesized for operation at 100 MHz. The column Delay describes the *maximum delay* which determines the highest operable frequency.

Table 4.1: Comparison of Hash Implementations at 100 Mhz

Design	**Dynamic Power**		**Leakage Power**		**Total Power**		**Circuit Area**		**Delay/ Speedup**	
	μW	**%**	**μW**	**%**	**μW**	**%**	**g.e.**	**%**	**ns**	**x**
NH	1093.9	100	28.1	100	1122.0	100	5291	100	9.92	1.0
PH	682.7	62	12.1	43	694.8	62	2356	45	1.35	7.4
PR	549.9	50	14.0	50	563.9	50	2537	48	1.35	7.4
WH	452.3	41	9.4	33	461.7	41	1701	32	1.35	7.4

WH consumes 41% of the dynamic power and 33% of the leakage power of NH while at the same time consuming only 32% of the area[2]. WH and NH need the same number of clock cycles to compute a hash value from the same input but WH can run 7.4 times faster than NH. We proved that WH provides the same level of security.

The dynamic power consumption of WH is 452.3 μW at 100 MHz. This is much higher than our aim of 20 μW. The CMOS power formula in Equation 2.1 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, however, the leakage power remains the same. This lower frequency is used in sensor node implementations [5]. Table 4.2 demonstrates that at low speeds the leakage power becomes the limiting factor for ultra-low-power implementations. WH can operate with as little as 11.6 μW. This is in the range of the power produced by a MEMS scavenger [89]. This table also shows how much energy each circuit consumes when it computes the hash for one 128-bit input as well as resulting energy per single bit. 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.

[2]g.e. stands for *gate equivalents.* It describes the area a circuit consumes in terms of two input NAND gates.

Table 4.2: Power and Energy Consumption of Hash Implementations at 500 kHz

Design	**Dynamic Power**		**Leakage Power**		**Total Power**		**Energy**		**Energy per bit**	
	μW	**%**	**μW**	**%**	**μW**	**%**	**nJ**	**%**	**pJ/bit**	**%**
NH	5.47	100	28.1	100	33.6	100	4.30	100	33.6	100
PH	3.41	62	12.1	43	15.5	46	1.98	46	15.5	46
PR	2.75	50	14.0	50	16.8	50	2.15	50	16.8	50
WH	2.26	41	9.4	33	**11.6**	35	1.49	35	11.6	35

4.5 Multi-Hashing and Toeplitz Construction

The newly introduced hash function family WH is a big improvement over NH in terms of power consumption. However, we can reduce the dominant leakage power consumption even further by utilizing Multi-Hashing and the well known Toeplitz approach. These techniques helps us to design a smaller circuit which therefore consumes less leakage power.

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 4.3.5), 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, [14] uses the technique of multi-hashing [123] 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 [84, 14, 73] 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 prove that the Toeplitz construction for WH can still achieve the desired result.

4.5.1 Toeplitz Construction

We introduce the hash function family $\mathsf{WH}^T[n,w,t]$ ("Toeplitz-WH") having 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 $M \in A$, $\mathsf{WH}^T_K(M)$ is defined as follows.

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

Theorem 5 *For any $w \geq 1$, $t \geq 1$, and any even $n \geq 2$, $\mathsf{WH}^T[n,w,t]$ 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 $\mathsf{WH}[n,w]$ and $\mathsf{WH}^T[n,w,t]$, respectively. Let M and M' be distinct members of the domain A with equal lengths. We are required to show

$$Pr[\mathsf{WH}^T_K(M) = \mathsf{WH}^T_K(M')] = 2^{-wt} \tag{4.5}$$

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.

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

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 (4.5) 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 $j \in \{1, \cdots, t\}$, $Pr[E_j \textit{ is true} \mid E_1, \cdots, E_{j-1} \textit{ are true}] = 2^{-w}$, 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

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

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

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

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

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

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,

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

□

4.5.2 WH with Toeplitz Construction

We have shown in Section 4.5.1 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 4.7 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.

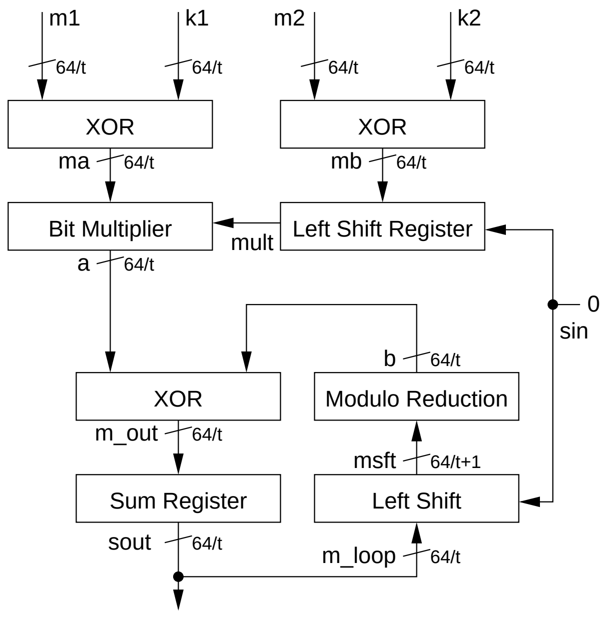

Figure 4.7: Detailed block diagram for WH datapath depending on Toeplitz parameter t

4.5.3 Analysis & Results of 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) using the TSMC 0.13 μm ASIC library and the Synopsys tools Design Compiler and Power Compiler. Table 4.3 shows the results for power, area, and delay for these hash implementations, synthesized for operation at 100 MHz .

Table 4.3: Comparison of Hash Implementations at 100 Mhz

Design	**Dynamic Power**		**Leakage Power**		**Total Power**		**Circuit Area**		**Delay / Speedup**	
	μW	%	μW	%	μW	%	**g.e.**	%	**ns**	**x**
WH-64	452.3	100	9.36	100	461.7	100	1701	100	1.35	1.0
WH-32	217.5	48	4.81	51	222.3	48	873	51	1.31	1.0
WH-16	126.2	28	2.32	25	128.5	28	460	27	0.76	1.8

It can be seen in Table 4.3 that the dynamic and leakage power consumptions as 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 2.1) 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 4.3 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.

4.5.4 Analysis of WH with Toeplitz

Table 4.4 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 4.4 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 stays 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 4.4: Comparison of Power and Energy Consumption

Design	**100 MHz**				**500 kHz**			
	P_{Dyn} μW	P_{Leak} μW	P μW	E nJ	P_{Dyn} μW	P_{Leak} μW	P μW	E nJ
WH-64	452.3	9.36	461.7	0.30	2.261	9.36	11.62	1.49
WH-32	217.5	4.81	222.3	0.28	1.087	4.81	5.90	1.51
WH-16	126.2	2.32	128.6	0.33	0.631	2.32	2.95	1.51

Design	**1 kHz**			
	P_{Dyn} nW	P_{Leak} μW	P μW	E nJ
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 4.4: 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 4.8 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 4.8 is logarithmic and only the powers of ten are shown.

The energy consumption is shown in Figure 4.9. The axes have a different orientation than in Figure 4.8 such that the frequency is decreasing towards the right and the area is decreasing towards the left. The frequency axis in Figure 4.9 is in logarithmic scale. Figure 4.9 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 4.8. 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 4.9, 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.

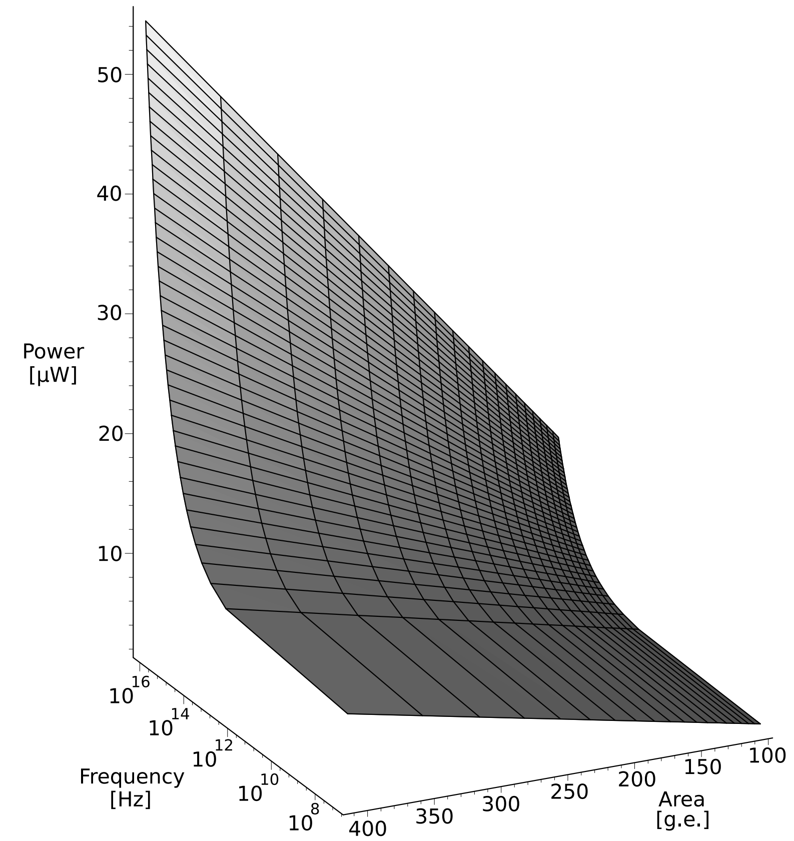

Figure 4.8: Power Consumption

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 maintain the runtime one may decide to

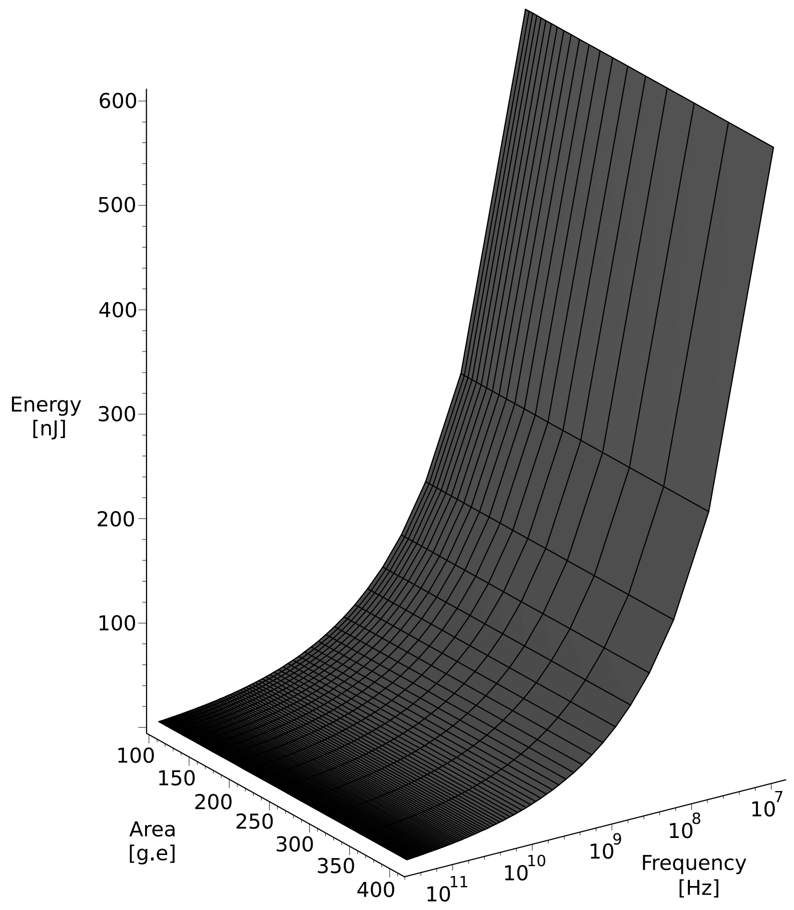

Figure 4.9: Energy Consumption

increase the operating frequency t times: $f'' = f\,t$. In this arrangement the dynamic 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

Table 4.5: Comparison of Power and Energy Consumption with $f' = f\,t$

Design	**f =100 MHz**					**f =500 kHz**				
	f' MHz	P_{Dyn} μW	P_{Leak} μW	P μW	E nJ	f' kHz	P_{Dyn} μW	P_{Leak} μW	P μW	E nJ
WH-64	100	452.3	9.36	461.7	0.30	500	2.261	9.36	11.62	1.49
WH-32	200	435.5	4.81	440.0	0.28	1000	2.175	4.81	7.00	0.89
WH-16	400	505.0	2.32	507.3	0.32	2000	2.525	2.32	4.84	0.62

Design	**f =1 kHz**				
	f' kHz	P_{Dyn} nW	P_{Leak} μW	P μW	E nJ
WH-64	1	4.523	9.36	9.37	599.5
WH-32	2	4.350	4.81	4.82	308.4
WH-16	4	5.050	2.32	2.32	148.5

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 4.5 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 f only. The actual clock frequency f' for WH-64 is equal to f, for WH-32 it is twice higher ($t = 2$) and for WH-16 it is 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 4.5 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.

4.6 Conclusion

In this chapter, we propose three variations on NH (the underlying hash function of UMAC), namely PH, PR and WH. Our main motivation 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.

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. Our implementation of WH consumes only $11.6\,\mu$W at 500 kHz.

The only way to reduce the leakage power even further 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.

By designing the new algorithms with efficiency guidelines in mind and applying optimization techniques, we achieved drastic power, energy and area savings. Our implementation of WH-16 consumes only $2.95\,\mu$W at 500 kHz and uses only 460 equivalent 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.

Chapter 5

Public Key Functions

In this chapter we present ultra-low power implementations of three inherently different public key algorithms: Rabin's Scheme, NtruEncrypt and Elliptic curve cryptography. Parts of this chapter were presented in [42] and [41]. The ECC point multiplication circuit was implemented by Erdinç Öztürk and is described in detail in [102] and [101]. Gunnar Gaubatz implemented the circuit for the "Star Multiplication" of NtruEncrypt.

5.1 Motivation

Most publication on wireless sensor network security seem to preclude that public key cryptography (PKC) is not feasible on severely power constrained devices, and therefore, revert to emulation of asymmetry using symmetric key techniques [108]. However, [117] shows that this is impossible for the service of broadcast authentication and therefore public key based schemes have to be used.

Most, if not all, implement cryptographic primitives in software on general purpose micro-controllers. While intuition might support this notion of infeasibility, we are not aware of any studies that have actually analyzed the cost of PKC in sensor networks, apart from [19] and our own publications [42] and [41].

Public key cryptography can enable services like broadcast authentication and tremendously simplify the implementation of many other security services and additionally reduce transmission power due to less protocol overhead. We show this in Chapter 7. Moreover, the capture of a single node would not compromise the entire network, since the nodes do not have to store any globally shared secrets. Our approach to overcome the difficulty in implementing PKC in sensor nodes is based on providing a custom-designed low-power co-processor that can be embedded in the node and that handles all of the compute-intensive tasks.

The challenge is to overcome the considerable computational complexity of standard public key encryption algorithms and make public key encryption possible on ultra-low power devices. Traditional schemes like RSA or ElGamal require considerable amounts of resources which in the past limited their use to large-scale platforms like networked servers and personal computers. Mobile equipment with less computational resources, such as cell phones, Personal Digital Assistants (PDAs) and pagers, therefore uses much more efficient elliptic curve based algorithms such as EC-DH and EC-DSA which execute considerably faster while preserving the same level of security [149]. The operands of EC-cryptosystems are much shorter than those in traditional schemes. Unfortunately the improved computational efficiency of ECC comes at the price of much more complex arithmetic primitives and a large number of temporary operands, whereas RSA or ElGamal require only one single arithmetic primitive and few operands. The heterogenous structure and larger storage requirements of ECC make it less scalable and more challenging for energy efficient low-power implementations.

5.2 Introduction

Rabin's Scheme, NtruEncrypt and Elliptic Curve Scalar Point Multiplication are inherently different algorithms. In order to be able to quantitatively compare them and their suitability for ultra-low power implementation we had to chose algorithm specific parameter sets that provide approximately the same level of security. In this section we talk about the rational behind our selection and briefly describe the algorithms.

5.2.1 Parameter Selection

When we talk about matching levels of security, we base our assumptions on the widely recognized analysis by Lenstra and Verheul [79]. They relate the selection of key sizes of various types of cryptosystems to the anticipated progress of cryptanalysis and cost of computation. They distinguish between key sizes of classical asymmetric systems (RSA, Rabin's Scheme, ElGamal, etc.), Subgroup Discrete Logarithm (DL) based schemes, and Elliptic Curve (EC) based systems. However, their analysis does not include a definition of equivalent security for a lattice based scheme like Ntru-Encrypt. For our purpose of finding parameters for NtruEncrypt, which offer a level of security comparable to the other two systems, we therefore refer to the analysis of Hoffstein, Silverman and Whyte [60].

While in practice certain classes of applications might require a higher level of security than others, we regard our designs simply as proof of concept and hence choose to implement them at a comparatively low level of security. It should, however, be relatively straightforward to estimate the cost of higher security level implementations based on the analysis that we give at the end of this Chapter. For Rabin's Scheme we selected a modulus of 512 bits, which according to Lenstra and Verheul [79] provides a security level of around 60 bits. Our ECC architecture performs arithmetic in a prime field of 100 bits in size, which provides a security level between 56 and 60 bits depending on the confidence level one puts into the assumption that no significant cryptanalytic progress has been made. In the case of NtruEncrypt we chose the system parameters as $(N, p, q) = (167, 3, 128)$, based on findings in [60], offering a security level of around 57 bits.

5.2.2 Rabin's Scheme

Rabin's Scheme was introduced in 1979 in [115]. It is based on the factorization problem of large numbers and is therefore similar to the security of RSA with the same sized modulus. Rabin's Scheme has asymmetric computational cost. The encryption operation is faster than decryption, which is comparable to RSA with similar parameters. Its asymmetry makes Rabin's Scheme an interesting choice for sensor network scenarios in which nodes and base stations have disparate computational capabilities. Below is a brief description of the Rabin's Scheme. For a detailed description and the mathematical proofs see [115][90].

Set-up

1. Choose two large random strong prime numbers.
2. Compute $n = p \cdot q$.
3. Pick a random number b for which $0 \leq b < n$.
4. The public key is (n, b), the private key is (p, q).

Encryption

1. Represent the message as an integer x for which $0 \leq x < n$
2. Compute the ciphertext y as $\mathrm{E}_{n,b}(x) \equiv x(x + b) \mod n$, as defined in [115]

Only the public key n, b is required for encryption. If we fix b to 0 then $\mathrm{E}_{n,b}(x)$ becomes a simple squaring operation $\mathrm{E}_n(x) = x^2 \mod n = y$. Rabin's Scheme requires only one squaring, whereas RSA requires several squarings and multiplications for encryption. Therefore encryption with Rabin's Scheme is several hundreds of times faster than RSA [121].

Decryption involves finding the roots of y. The decryption function is $\mathrm{D}_n(y) \equiv \sqrt{y} \mod n = \{x_1, x_2, x_3, x_4\}$ and yields four results. In order to determine the correct solution, sufficient redundancy has to be included in x. Certain simplifications are possible if $p \equiv q \equiv 3 \mod 4$. We would like to point the interested reader to [90] for a complete description of these algorithms. We did not pursue a hardware implementation of the decryption function.

5.2.3 The NtruEncrypt Public Key Cryptosystem

NtruEncrypt is a relatively new cryptosystem that claims to be highly efficient and particularly suitable for embedded applications such as smart cards or RFID tags, while providing a level of security comparable to that of other established schemes, in particular RSA. While it has not yet received the same level of scrutiny for establishing its resistance to cryptanalysis, there is evidence for efficiency in the simplicity of its underlying arithmetic. In this section we briefly describe the basic setup of NtruEncrypt and its operations. For more in-depth descriptions of the mathematical properties of NtruEncrypt we refer to [59, 57].

NtruEncrypt is based on arithmetic in a polynomial ring $R = \mathbb{Z}(x)/((x^N - 1), q)$ set up by the parameter set (N, p, q) with the following properties:

- All elements of the ring are polynomials of degree at most $N - 1$, where N is prime.

- Polynomial coefficients are reduced either mod p or mod q, where p and q are relatively prime integers or polynomials.
- p is considerably smaller than q, which lies between $N/2$ and N.
- All polynomials are univariate over the variable x.

Multiplication in the ring R is sometimes referred to as "Star Multiplication" based on use of an asterisk $\circledast$ as the operator symbol. It can be best described as the discrete convolution product of two vectors, where the coefficients of the polynomials form vectors in the following way:

$$\begin{aligned} a(x) &= a_0 + a_1 x + a_2 x^2 + \ldots + a_{N-1} x^{N-1} \\ &= (a_0, a_1, a_2, \ldots, a_{N-1}) \\ b(x) &= (b_0, b_1, b_2, \ldots, b_{N-1}) \\ c(x) &= (c_0, c_1, c_2, \ldots, c_{N-1}) \end{aligned}$$

Then the coefficients c_k of $c(x) = a(x) \circledast b(x) \bmod q, p$ are each computed as

$$c_k = \sum_{i+j=k \bmod N} a_i b_j$$

The modulus for reduction of each coefficient c_k of the resulting polynomial is either q for Key Generation and Encryption, or p for Decryption, as briefly described below. A thorough description of these procedures along with an initial security analysis can be found in [59].

Set-up The following steps generate the *private key* $f(x)$:

1. Choose a random polynomial $F(x)$ from the ring R. $F(x)$ should have small coefficients, i.e. either binary from the set $\{0, 1\}$ (if $p = 2$) or ternary from $\{-1, 0, 1\}$ (if $p = 3$ or $p = x + 2$ [57, 8]).

2. Let $f(x) = 1 + pF(x)$ [1].

The *public key* $h(x)$ is derived from $f(x)$ in the following way:

1. As before, choose a random polynomial $g(x)$ from R.
2. Compute the inverse $f^{-1}(x) \pmod{q}$.
3. Compute the public key as $h(x) = g(x) \circledast f^{-1}(x) \pmod{q}$.

Encryption

1. Encode the plaintext message into a polynomial $m(x)$ with coefficients from either $\{0, 1\}$ or $\{-1, 0, 1\}$.
2. Choose a random polynomial $\phi(x)$ from R as above.
3. Compute the ciphertext polynomial $c(x) = p\phi(x) \circledast h(x) + m(x) \pmod{q}$.

Decryption

1. Use the private key $f(x)$ to compute the message polynomial $m'(x) = c(x) \circledast f(x) \pmod{p}$.
2. Map the coefficients of the message polynomial to plaintext bits.

5.2.4 Elliptic Curve Cryptography

Elliptic Curve Cryptography (ECC) is the collective term for a multitude of different asymmetric cryptographic key exchange and agreement protocols, e.g. Diffie-Hellman

[1] It is not strictly necessary to construct $f(x)$ in this way, but it is recommended in order to decrease the decryption failure rate. It is important, however, that $f(x)$ be invertible $(\text{mod} p)$ and $(\text{mod} q)$.

Key Exchange with Elliptic Curves (ECDH), Elliptic Curve Digital Signature Algorithm (ECDSA), Menezes-Vanstone Elliptic Curve Cryptosystem (ECMV), and many more. Scalar point multiplication serves as the basic building block of these and is the computationally most expensive operation. Different types of finite fields can be used for the construction of elliptic curve groups. The most common ones are Galois Fields with prime characteristic or binary extension fields, e.g. GF(p) and GF(2^k). Efficient arithmetic in these fields is the key to low-power implementations of ECC in hardware.

For the purpose of establishing the feasibility of an ECC based public key primitive in a pervasive security context, we selected ECMV for data encryption. Below is a brief description of ECMV. A more detailed description can be found in [134].

Set-up

1. Choose an elliptic curve E : $y^2 \equiv x^3 + \alpha \cdot x + b \mod p$.
2. Choose a primitive element $\alpha = (x_\alpha, y_\alpha) \in E$.
3. Pick a random integer $\alpha \in \{2, 3, \ldots, \#E - 1\}$.
4. Compute $a \cdot \alpha = \beta = (x_\beta, y_\beta)$.
5. The public key is (E, p, α, β), the private key is (a).

Encryption

1. Pick a random $k \in \{2, 3, \ldots, \#E - 1\}$.
2. Compute $k \cdot \beta = (c_1, c_2)$.
3. Encrypt $e_{E,p,\alpha,\beta}(x, k) = (Y_0, Y_1, Y_2)$ where $Y_0 = k \cdot \alpha$, $Y_1 = c_1 \cdot x_1 \mod p$, and $Y_2 = c_2 \cdot x_2 \mod p$.

Decryption

1. Compute $a \cdot Y_0 = (c_1, c_2)$.

2. Decrypt $d_a(Y_0, Y_1, Y_2) = (Y_1 \cdot c_1^{-1} \mod p, Y_2 \cdot c_2^{-1} \mod p) = (x_1, x_2)$.

5.3 Implementations

We have made for following assumptions for our implementations:

- Depending on the exact application scenario it might be possible to fix the public key to a constant value. This is extremely beneficial for ultra-low power implementation, since the key can be embedded statically and does not require costly storage elements. In our implementations the public key is either hard-wired or realized as a look-up table in combinational logic.

- As stated in the introduction, we only consider the encryption operation of both systems. The purpose of these implementations is to show that public key cryptography is computationally feasible on ultra-low power devices.

5.3.1 Rabin's Scheme

We have shown in Section 5.2.2 that the basic function for encryption in Rabin's Scheme is a simple squaring operation $E_n(x) = x^2 \mod n$, if we set $b = 0$.

Squarers are a special form of multiplier. While any multiplier can be used to compute the square of a number, special-purpose squarers usually require significantly less hardware and are faster [103] by exploiting the symmetry of the squaring operation. Table 5.1 shows in the upper part how the multiplication of an unsigned integer multiplied with itself is computed by a multiplier. The lower part presents the simplified squaring algorithm.

Table 5.1: Multiplication vs. Squaring using Integers

				x_3	x_2	x_1	x_0
				x_3	x_2	x_1	x_0
				x_3x_0	x_2x_0	x_1x_0	x_0x_0
$+$			x_3x_1	x_2x_1	x_1x_1	x_0x_1	
$+$		x_3x_2	x_2x_2	x_1x_2	x_0x_2		
$+$	x_3x_3	x_2x_3	x_1x_3	x_0x_3			
	x_3x_2	x_3x_1	x_3x_0	x_2x_0	x_1x_0	0	x_0
$+$	x_3		x_2x_1		x_1		
$+$			x_2				

Squarers can be implemented in many ways. As our main concern is to conserve power we chose a bit-serial approach. The main advantage of a bit-serial design is that it minimizes the number of gates and reduces wire lengths—all factors that are of concern with regards to the circuit's power consumption. Bit-serial implementations can be almost competitive with complex designs with regard to speed if they are driven at a high clock rate. A bit-serial squarer which uses the Least Significant Bit (LSB) first method was presented in [62].

The bit-serial approach is ideal for modular reduction. Using the most significant bit (MSB) first method, modular reduction can be performed elegantly after each partial product addition. Table 5.2 shows how this would work with an optimized squarer. Each dashed line indicates the end of one clock cycle. After each addition is completed the result gets modulo reduced. Modulo reduction works by checking for a carry on the MSB and if it is set then adding the two's complement of the modulus to the result. The difficulty in the approach shown in Table 5.2 is the generation of the multiplicand sequence ($x_3x_2x_1x_0$, $0x_2x_1x_0$, $00x_1x_0$, $000x_0$) which requires an extra 512-bit register.

This is very expensive in terms of area and leakage power as each flip-flop is the

Table 5.2: Squaring with Modulo Reduction

				x_3	x_2	x_1	x_0
				x_3	x_2	x_1	x_0
+	x_3						
+	x_3x_2	x_3x_1	x_3x_0				
+			x_2				
+			x_2x_1	x_2x_0			
+					x_1		
+					x_1x_0		
+							x_0
Modulo Reduction				No Reduction			

equivalent of 6 gates. Therefore, we implemented the squarer as a bit serial modular multiplier where multiplicand and multiplier are hard-wired to the same input. All 512 bits of input are available in parallel at the same time. As a multiplier does not take advantage of the symmetry in squaring we expect it to consume more switching power. However, due to its smaller footprint the leakage power is also greatly reduced. At the low clock frequencies commonly encountered in sensor nodes, the influence of leakage power is the dominant part. An additional advantage of this approach is that this unit can easily be converted to a full multiplier for an implementation of RSA or a similar algorithm.

Data Path Figure 5.1 shows the architecture of our squarer. It is a standard bit serial multiplier design comprised of a Left Shift Register, a Bit Multiplier, a Left Shift unit, and the main units Adder and Sum Register. In order to perform modular multiplication we added two multiplexers which toggle the input of the adder between the next partial product and the 2's complement of the modulus n (reduction). The control logic determines whether a reduction operation is necessary after an addition. Since we are using the same adder for both functions, the number of clock cycles

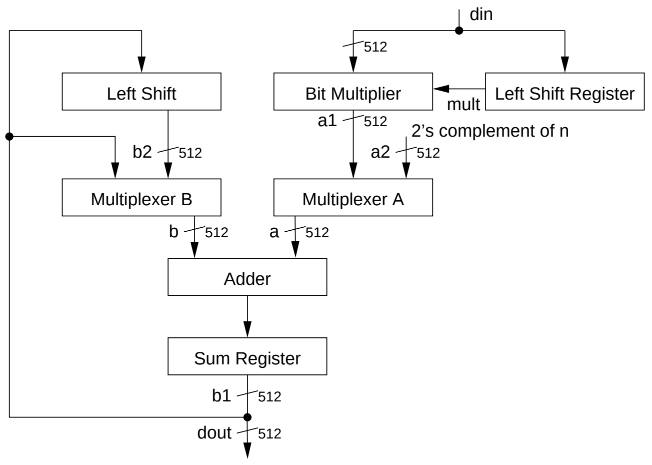

Figure 5.1: Block Diagram for Rabin's Scheme

needed for one squaring is data dependent and at most 1024.

Adder The most complex part of the squarer is the Adder. There are two basic adder designs that are suitable for a low power implementation, namely carry-save adder and ripple-carry adder. The ripple carry adder consists of one half adder and 511 full adders which is the bare minimum hardware for a 512 bit adder. Therefore the ripple carry adder has the least leakage power consumption of all alternatives. However as each input bit might generate a carry and all carries are propagated immediately this adder also has the longest delay. The propagation of carries causes glitches which in turn cause a very high dynamic power consumption.

A carry-save adder on the other hand propagates carries only by one position, hence there are no glitches, resulting in insignificant amounts of delay and dynamic power consumption. Its disadvantage is that the result is kept in redundant carry-save representation which requires 512 additional flip-flops for the storage of the carry bits. This in turn causes a higher consumption of leakage power. Since partial products and complements of the modulus can be accumulated in redundant form, the final non-redundant result needs to be computed only at the very end of the multiplication which takes 512 additional clock cycles.

Neither of both approaches seems optimal for this implementation, so we tried to strike a balance between power and speed. For our adder we are using a ripple-carry adder and insert a carry-save bit on every 8th bit position. Hence the carries ripple for a maximum of 8 bits causing some glitching but significantly less than a full ripple-carry adder would. The dynamic power consumption is therefore much lower than for a full ripple-carry adder. This adder also needs only 64 additional flip-flops to store the carry bits, which is 448 flip-flops less than necessary for a full carry-save adder. This approach, however, introduces a new difficulty. After adding a partial product to the sum, the result has to be shifted. This would misalign the saved carry bits [2]. Hence, carry bits need to be re-aligned before shifting the sum. This is done by adding the carry bits to the sum in the appropriate position and saving the carry bits at the new position. The cost for this is a 512 bit multiplexer, 512 additional clock cycles and a slightly more complex control logic.

Control Logic The Control logic is comprised of two state machines and one counter. The counter is implemented as a Linear Feedback Shift Register (LFSR) and "counts" up to 512. LFSRs have reduced switching activity and are faster than regular counters, hence reducing the effects on the critical path delay. Furthermore it is clock gated and can be reset. The counter is used to count all the multiplication steps and also to count the worst case number of steps necessary to ripple all 64 carry-save flip-flops. The main state machine of this control logic keeps track of the overall operation of the circuit. Its states are:

1. *Load:* Loading a new operand into the left shift register.

2. *Multiply:* Multiplying the operand with itself.

[2]This problem does not occur when a full carry-save adder is being used as there is one carry bit associated with every bit position

3. *Ripple:* Compute the final carry free result.

4. *Done:* The output of the circuit has a valid result.

The second state machine takes care of arithmetic operations of the circuit. Furthermore it is responsible for the clock gating of the counter and the left shift register (see Figure 5.1). This state machine is used during the *Multiply* and *Ripple* states of the main state machine and uses the following states:

1. *Add:* Add the partial product to the sum.

2. *Shift:* Realign the saved carries and shift the result by one bit.

3. *Modulo Reduce:* Add the 2's complement to the result.

5.3.2 NtruEncrypt and NtruSign

The basis for our low-power NtruEncrypt architecture is the multiplication operation in the ring R, which basically is a cyclic convolution of two polynomials of the same degree N. This operation is sometimes also referred to as "Star Multiplication", due to the use of an asterisk ('$\circledast$') as the operator symbol. The two polynomials' coefficients are of different size, one is reduced modulo q, the other modulo p. The main goal of our architecture is to be as energy efficient as possible. For now we will only consider a scenario in which a sensor node encrypts a message. This allows us to make the following observations which are helpful for an energy efficient architecture.

- Encryption involves loading a random polynomial $\phi(x)$ and expanding it to a pseudo one-time pad using the "star multiplication" $p\phi(x) \circledast h(x) \bmod q$. The message polynomial $m(x)$ is encrypted by adding the pad to the message modulo q, coefficient by coefficient. The expansion of the pad does not have to be computed at once, it is sufficient to compute one coefficient at a time, i.e. the circuit can be stalled until there is another message coefficient.

- As mentioned at the beginning of this section, the public key of the node $h(x)$ is constant and embedded in the device. Since p is also constant, we can store a pre-scaled version of the public key $h'(x) = ph(x) \bmod q$. Thus we only need to compute $c(x) = \phi(x) \circledast h'(x) + m(x) \bmod q$.

- The operands involved in the star multiplication have coefficients of different word sizes. The public key $h(x)$ is a polynomial with coefficients mod$q$, i.e. with large word size. This leaves the smaller word size mod$p$ for coefficients of the random polynomial $\phi(x)$. This directly translates into fewer storage elements (i.e. flip-flops). For our choice of NtruEncrypt parameters $(N, p, q) = (167, 3, 128)$, the coefficients of $\phi(x)$ will be encoded using two bits. Since the public key is constant and realized as a look-up table, only $2N = 334$ bits of storage are required as opposed to $N\lceil \log_2 q \rceil = 1169$.

- We assume that we have a good source of random bits available for generation of the random polynomial $\phi(x)$. In this dissertation we focus on the computational aspects of cryptographic algorithms only, and therefore random number generation falls outside of the scope. For information on a compact implementation of an RNG based on digital artefacts requiring only a few hundred gates we refer to [34].

NtruEncrypt Data Path The algorithm implementing cyclic convolution consists of two nested loops. The outer loop iterates over all N coefficients of the result. The inner loop computes the coefficient by accumulating products of the form $a_i b_j$, with index i increasing and j decreasing mod$N$. The three major building blocks comprising the data path of the circuit—public key look-up table (LUT), arithmetic units and circular buffer—are illustrated in Figure 5.2. The public key look-up table, mapping the index of the coefficient to its value, is realized in combinational logic that lends itself to optimization through the synthesis tool. The circular buffer consists of $2N$ bits of storage elements containing the coefficients of the random polynomial $\phi(x)$. Data enters the buffer through a multiplexer which connects the two ends of the buffer and forms a ring. Both, public key LUT and circular buffer, feed into the arithmetic units (AUs) which multiply and accumulate the operands.

The smallest version of the circuit implements only a single AU. Yet, the architecture allows the implementor to scale up the number k of parallel AUs relatively easily, with minimal impact on the other elements of the design. Section 5.4.4 elaborates further on NtruEncrypt's inherent scalability.

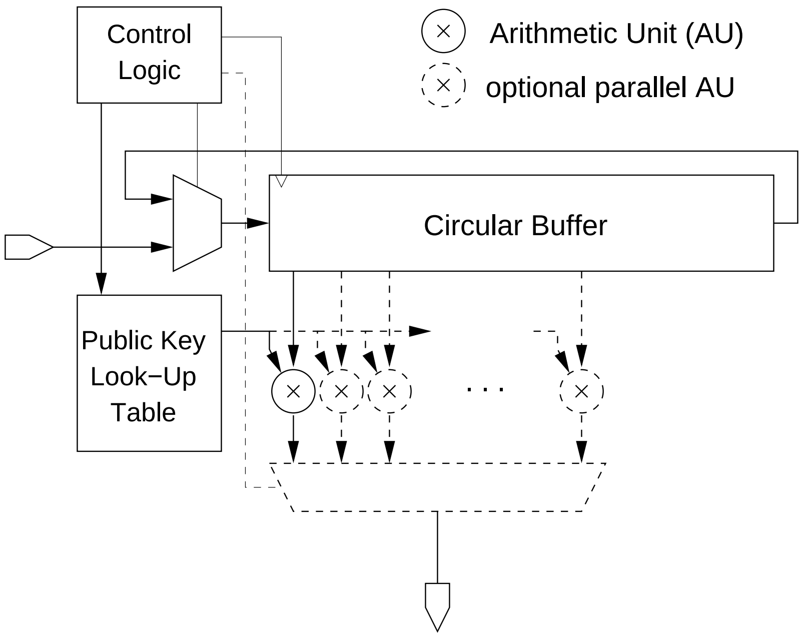

Figure 5.2: NtruEncrypt block diagram

Arithmetic Unit An AU consists of a partial product generator, a carry-save adder (CSA) and a register. For any long operand a and short operand b the partial product generator will compute $ab \bmod q$. The partial product generator has a relatively simple structure since the short operand uses only two bits. The coefficient values $\{-1, 0, 1, 2\}$ are encoded in two bits as $\{11, 00, 01, 10\}$ respectively. The final partial product is selected from the set $\{-a, 0, a, 2a\}$ using a multiplexer. The value $-a$ is computed efficiently by creating the ones-complement with inverters and setting the incoming carry bit of the CSA to one. By choosing $p = 3$ and $q = 128$ the modular reduction of the intermediate result $c = \sum a_i b_j \bmod q$ comes essentially for free through simple truncation of bits at positions $\geq \log_2 q = 7$. Once all convolution products and the message coefficient have been accumulated, the CSA is clocked for seven more cycles with zero input. This propagates all in-flight carry bits and produces a non-redundant result.

Control Logic The control logic is designed to be as simple as possible in order to avoid being the bottleneck in terms of power consumption. The four states of the finite state machine are encoded binary and not one-hot, in order to use the minimum number of state registers. These states are

1. *Load/Run*: Load coefficients of $\phi(x)$ and compute c_0 / Run the computations of all other coefficients of $c(x)$ (N clock cycles)
2. *Add*: Add the coefficient m_i of the message polynomial $m(x)$ to the partial convolution product (1 clock cycle)
3. *Propagate*: Propagate all in-flight carries of the CSA to obtain a non-redundant final sum (6 clock cycles)
4. *Done*: Signal completion of computation for a given coefficient and eventually the whole ciphertext polynomial (1 clock cycle)

The two nested counters needed for keeping track of coefficients in the inner and outer loop of the algorithm are implemented as Linear Feedback Shift Registers (LFSR) for reduced switching activity. Furthermore, clock gating is used extensively whenever possible, to avoid any unnecessary switching activity and reduce parasitic wire capacitance.

The inner loop counter provides the index value for looking up coefficients in the public key LUT as well as start and end triggers for the control logic, i.e. when to add the message coefficient $m_i(x)$ to the result and when to count out six additional clock cycles during which the CSA propagates carry bits. The outer loop counter is initialized through global reset at the beginning of the operation and increases its value whenever computation of a coefficient has been completed. It also keeps track of the number of rounds (one per k coefficients) that have been computed and upon completion raises a signal.

In the case of only a single AU, i.e. $k = 1$, each round of computation takes $N + 8$ clock cycles to complete, with one coefficient per round. The eight additional clock cycles are necessary for addition of the message coefficient and propagation of carries in the carry-save adder. The total number of clock cycles for a full polynomial multiplication of N coefficients therefore takes $29,225$ clock cycles ($N = 167$). If k AUs are computing coefficients in parallel, the rounds overlap partially and the number of clock cycles amounts to $(N + 8)(\lceil N/k \rceil) + k - 1$. For a high degree of parallelization k the number of clock cycles can thus be reduced dramatically, i.e. to only 433 cycles for $k = 84$.

Between two rounds the data in the circular buffer is rotated in order to be correctly aligned for the next round of k coefficients.

5.3.3 Elliptic Curve Architecture

Our ECC scalar point multiplication implementation is based upon efficient arithmetic in prime fields with moduli of special form. We employ a novel modulus scaling technique that yields a composite modulus of the form $m = s \cdot p = 2^k + 1$, where the scaling factor s is very small. This leads to a very efficient method for almost modular reduction, where intermediate results are only reduced to within a small multiple of m. Only after the final step of the scalar point multiplication the result is fully reduced by means of repeated subtraction. For our implementation we choose an an elliptic curve given by the equation $y^2 = x^3 + ax + b$, defined over the field GF($(2^{101} + 1)/3$) and make use of the special scaled modulus $m = 2^{101} + 1$. A scaling factor of $s = 3$ has been chosen, because it is sufficiently small so that the sizes of registers in the design increases only marginally. Furthermore, the use of these moduli of special form allows us to use the very efficient Algorithm X for modular inversion of Mersenne primes that was first proposed by Thomas et al. [140]. With a slight modification the algorithm can be made to also work with scaled moduli $m = s \cdot p$ where $\gcd(a, m) \neq 1$. For our implementation we chose affine coordinates which yield faster inversion and they require less storage space than projective coordinates.

Arithmetic Core Figure 5.3 illustrates the data path of the ECC arithmetic core which implements all arithmetic primitives such as addition, subtraction, multiplication and division (inversion) by extensively reusing hardware components.

Adder The main functional block on both sides of the data path is the adder. All arithmetic is implemented in carry-save form which ensures a moderate amount of switching activity in contrast to a full carry propagation adder. The downside to this approach is the increased cost of storage for the redundant representation and the addition, shift, rotation and comparison operations become more cumbersome.

Control Logic The control logic determines which function the arithmetic core performs by switching the multiplexers appropriately. It consists of a state machine with 15 states to implement the inversion algorithm. The main states are:

1. *Initialize* Loads the appropriate initial values into the registers.
2. *Shift Right* Both, the sum and the carries have to be shifted.

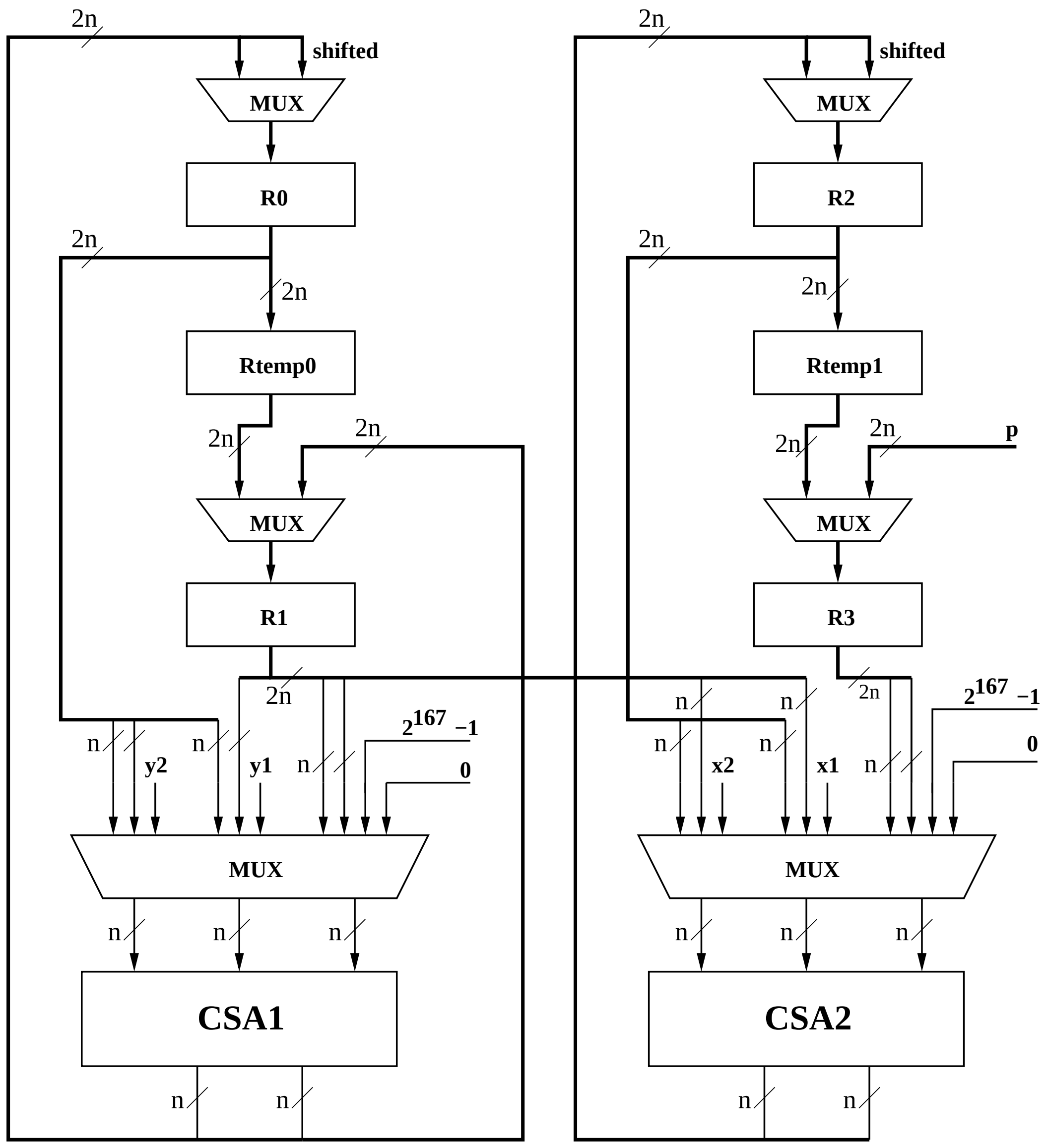

Figure 5.3: Block Diagram for the Arithmetic Unit for ECC

3. *Add* The addition of two numbers in carry save notation takes two clock cycles in our architecture as the carry save adder has only three inputs and a number in carry save notation consists of two parts, so overall 4 parts have to be added.

Comparator The modified version of Algorithm X for inversion with respect to a scaled modulus requires a comparison of the intermediate result with the constant scale factor s. A traditional logical OR tree-based approach would be prohibitively costly in the number of gates required and would also introduce a significant amount of delay. Our architecture employs a wired-or using tristate buffers as shown in Figure 5.4 which is described in [101].

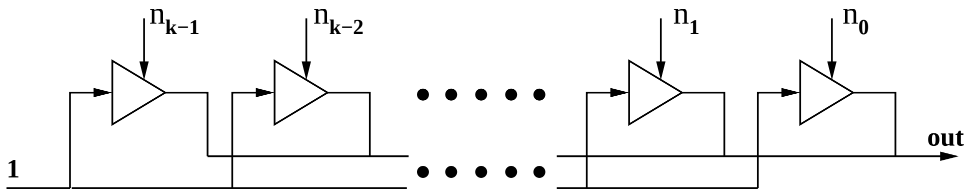

Figure 5.4: Comparator unit built using tri-state buffers

5.4 Analysis

In this section we analyze the proposed architectures according to various metrics of interest to ultra low-power applications such as wireless sensor nodes. Since the three public key algorithms and their hardware architectures are distinctly different from each other a direct comparison is difficult. We alleviate this situation by fixing system parameters to values that match security levels of all three systems as closely as possible, as mentioned in Section 5.2.1.

5.4.1 Rabin's Scheme

The main concern driving our low-power implementation of Rabin's Scheme is its storage requirement. Many well known techniques for optimizing a modular squarer require either more circuitry or more storage elements. At our targeted clock frequency of 500 kHz the static power consumption is dominant and therefore has to be minimized. Hence, we built a squarer as a bit-serial multiplier, operating on the entire width of the 512 bit multiplicand and on a single bit of the multiplier at a time. In order to conserve area we use the same adder for accumulating the partial products, modulo reducing the results, and re-aligning the carry bits before each shift. This approach consumes a chip area of less than $17,000$ gates with its accompanying static power consumption of $117.50\mu W$. The dynamic power consumption at 500 kHz is $30.68\mu W$ resulting in a total average power consumption of $148.18\mu W$. Table 5.3 shows these results and also breaks down the area into area used for combinational logic (Cmb.) and storage (Reg.). A breakdown of the power consumption by functional blocks reveals an interesting relationship. The adder consumes 39.8% of the power but only 24.8% of the area, whereas all storage elements combined consume 37.8% of the total power consumption but 55.6% of the area. The disproportional power consumption of the adder is due to its much higher dynamic power consumption, even at the low frequency of 500 kHz. It is also interesting to note that the power consumption of the complex control logic for this circuit is negligible at 1.6%. The column "other" in Table 5.3 comprises the multiplexer and the Bit Multiplier as shown in Figure 5.1.

Table 5.3: Rabin's Scheme area and power consumption by function at 500 kHz

Blocks	**Power**				**Area**			
	P_{Dyn}	P_{Leak}	P_{Total}		**Cmb.**	**Reg.**	**Total**	
	μ**W**	μ**W**	μ**W**	**%**	**g.e.**	**g.e.**	**g.e.**	**%**
Storage	6.82	49.14	55.96	37.8	1,547	7,746	9,292	55.6
Adder	16.97	42.01	58.98	39.8	4,146	0	4,146	24.8
Control	1.00	1.33	2.33	1.6	77	105	182	1.1
Other	5.88	25.00	30.88	20.8	3,103	0	3,103	18.6
Total	30.68	117.50	148.18	100.0	8,874	7,851	16,726	100.0

The high dynamic power consumption of the adder at 500 kHz gets even more amplified when we increase the clock frequency. Figure 5.5 shows the relative power consumption of the functional parts of our Rabin's Scheme implementation over clock frequency from 500 kHz till 100 MHz. At high frequencies the adder dominates the power consumption with 55% while the relative power consumption of the storage units declines to 23%. The control logic shows only a moderate increase. Recall that the adder consists of 8-bit ripple carry adders which exhibit glitching. At high frequencies, where the dynamic power consumption is dominant, a full carry save adder would have yielded in a lower power consumption.

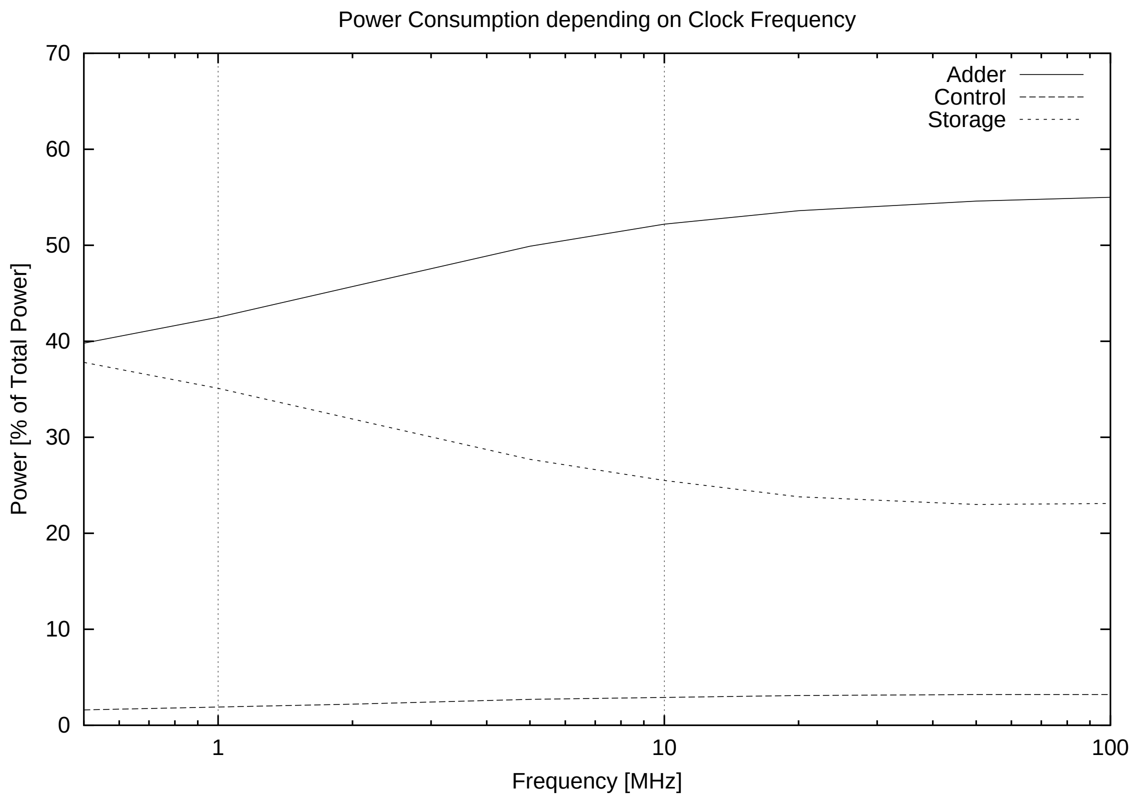

Figure 5.5: Rabin's Scheme: Power consumption of parts in % of total based on frequency

5.4.2 NtruEncrypt

The architecture implementing NtruEncrypt was developed with scalability and potential for parallelization explicitly in mind. The hardware friendly arithmetic that underlies the NTRU system greatly facilitated this process, since the computation of each of the ciphertext polynomial's coefficients is independent of one another. Furthermore the use of a circular buffer rotating the coefficients of $\phi(x)$ allows multiple parallel accesses by tapping the buffer at different positions. All AUs use the same public key coefficient in each iteration, so there is no need to have multiple look-up tables. Parallel computation of the result can therefore be achieved by simply replicating the AU and slightly modifying the control logic. Figure 5.2 in Section 5.3.2 shows a block diagram of the architecture, where solid lines represent the minimal configuration. Additional circuit elements for parallelization such as multiple AUs and a multiplexer are drawn using dashed lines.

Table 5.4: NtruEncrypt area and power consumption by function at 500 kHz

Blocks	**Power**				**Area**			
	P_{Dyn}	P_{Leak}	P_{Total}		**Cmb.**	**Reg.**	**Total**	
	μ**W**	μ**W**	μ**W**	%	**g.e.**	**g.e.**	**g.e.**	%
Circular Buffer	3.37	11.33	14.70	76.8	6	2,120	2,126	74.6
Public Key LUT	0.33	1.59	1.92	10.0	391	0	391	13.7
Control Logic	0.18	1.10	1.28	6.7	47	126	172	6.0
Arithmetic Unit	0.14	1.06	1.20	6.3	74	82	157	5.5
Total	19.13	4.03	15.10	100.0	523	2,327	2,850	100.0

We can make an interesting observation at this point by looking at the contribution of each functional block to the overall circuit area (Table 5.4). The most significant contribution to the overall power consumption is made by the circular buffer (77%), yet its size is only proportional to the system parameter N and independent of the degree of parallelization k. Similarly, the public key look-up table (10%) is independent of k as well. However, an arithmetic unit contributes the least amount of only 6.3%.

The cost for an implementation with only a single arithmetic unit is therefore relatively high compared to a more parallelized variant. The small cost of adding arithmetic units, on the other hand, allows for a high degree of parallelization. This level of scalability is advantageous when it comes to achieving optimal energy efficiency. In the following analysis we therefore also consider performance estimates for a highly parallelized ($k = 84$) variant of our NtruEncrypt architecture, based on data obtained from simulation of the digit serial implementation ($k = 1$).

Our implementation of the encryption function of NtruEncrypt takes up a chip area of less than 3000 equivalent gates for $k = 1$. This figure includes the storage elements for the random polynomial $\phi(x)$ and the combinational look-up table of the scaled public key $h'(x)$. Gate level power simulation indicates an average power consumption of under $20\mu W$ at a clock frequency of $500kHz$, close to the amount of static leakage power (see Table 5.6).

5.4.3 Elliptic Curve Architecture

The architecture occupies a chip area equivalent to 18,720 gates and consumes just under 400 μW of power at a clock frequency of 500 kHz (see Table 5.5). The dynamic power consumption of this circuit is three times as high as its leakage power at 500 kHz. This is somewhat surprising as we stated that at these low frequencies the leakage power is usually dominant. In addition to this, all adders in this circuit are carry save adders which do not cause glitching and therefor exhibit less dynamic power consumption but a larger leakage power consumption than for example ripple carry adders. The detailed breakdown in Table 5.5 sheds some light on this mystery. It shows that the main power consumer are the multiplexers which channel the data through the adders and registers. Also the storage units have a higher dynamic power consumption than leakage because of the frequent input changes.

Table 5.5: ECC area and power consumption at 500 kHz

Blocks	**Power**				**Area**			
	P_{Dyn}	P_{Leak}	P_{Total}		**Cmb.**	**Reg.**	**Total**	
	μ**W**	μ**W**	μ**W**	%	**g.e.**	**g.e.**	**g.e.**	%
Storage	93.68	43.73	137.41	34.8	702	7,852	8,554	45.7
Adder	13.40	7.40	20.80	5.3	1,496	0	1,496	8.0
Multiplexer	146.08	21.61	167.69	42.5	5,175	0	5,175	27.6
Control	17.78	3.98	21.76	5.5	377	190	566	3.0
Other	22.07	25.28	47.35	12.0	2,929	0	2,929	15.6
Total	292.58	101.82	394.40	100.0	10,679	8,042	18,720	100.0

5.4.4 Comparison

Table 5.6 shows a direct comparison between Rabin's Scheme, both variants of NtruEncrypt, and ECC. The architectures of Rabin's Scheme, the simple variant of NtruEncrypt, and ECC were intended to achieve the least possible power consumption given the available standard cell library, without necessarily reaching optimal energy efficiency. After an initial analysis of the architectural differences we decided to include estimates for a highly parallelized variant of NtruEncrypt in the third column of the comparison. The degree of parallelization $k = 84$ was chosen in a way that the area footprint roughly matches that of Rabin's Scheme and ECC, and secondly that $\lceil N/k \rceil - N/k$ is as small as possible. This is to divide the number of coefficients N in a way that utilization of the AUs is high during the last round of computation. The delay shows the number of clock cycles needed for one encryption operation. For ECMV, two point multiplications are required, i.e., the ECC circuitry has to be run on two different sets of inputs. The column "Throughput" takes this into account and presents a normalized value.

Table 5.6: Comparison of Encryption with Rabin's Scheme, NtruEncrypt, and ECC

	Rabin	Ntru (k=1)	Ntru (k=84)	ECMV
Equivalent security	60 *bits*	57 *bits*	57 *bits*	61 *bits*
Area [g.e.]	16,726	2,850	16,200	18,720
- combinational	8,875	523	7,000	10,679
- storage elements	7,851	2,327	9,200	8,041
Delay (avg. # cycles)	1,440	29,225	433	408,850
Avg. power [μW]	148.18	19.13	118.7	394.4
- static [μW]	117.50 (79.3%)	15.10 (78.9%)	103.06 (86.8%)	101.82 (25.8%)
- dynamic [μW]	30.68 (20.7%)	4.03 (21.1%)	15.64 (13.2%)	292.58 (74.2%)
Energy [nJ]	426.76	1,118.15	102.79	322,501.88
- per bit [pJ]	833.5	4,235.4	389.4	1,612,509.40
min input [bits]	512	264	264	200
Throughput [kbits/s]	177.78	4.52	304.85	0.24

Area Rabin's Scheme takes up almost six times the area of simple NtruEncrypt with a single AU. On the other hand it also has the advantage of performing almost forty times faster. This is to be expected due to its large operands and full-word arithmetic. If, however, the absolute area and power requirements are the limiting factor, it might not be flexible enough. Our ECC architecture exhibits the largest area requirements, even though it is only 10% larger than Rabin's Scheme, and it is the slowest, with Rabin's Scheme performing 740 times faster and simple NtruEncrypt almost 19 times faster. Also, our estimates for the parallelized variant of NtruEncrypt indicate that it outperforms Rabin's Scheme by nearly factor two using the same area footprint.

Power Consumption From the figures in Table 5.6 it is evident that static leakage power is the main culprit for the relatively high energy consumption of both Rabin's

Scheme and NtruEncrypt implementations. However, this is not the case for ECC. We would like to stress the fact that leakage power is highly technology dependent and that the ASIC standard cell library we use is not optimized for low power design. The dynamic power consumption of an architecture, on the other hand, is proportional to its switching activity. It therefore makes sense to differentiate between these two influences if we want to compare the relative merits of one architecture over the other, independently of the process technology. It turns out that dynamic power consumption in Rabin's Scheme is nearly twice as high as in the parallel NtruEncrypt's case. The dynamic power consumption of ECC is 10 times higher than that of Rabin's Scheme. A large portion of its dynamic power is caused by the multiplexers as we have shown in Section 5.4.3. The leakage power of our ECC implementation is similar to the ones of Rabin's Scheme and the parallel version of NtruEncrypt which is to be expected as all three are of similar size.

Figure 5.6 contains a graph with plots of power consumption over clock frequency for Rabin's Scheme, the two NtruEncrypt variants, and ECC. Common to Rabin's Scheme and NtruEncrypt is the dominance of static power consumption at low frequencies, while at high clock frequencies dynamic power consumption takes over. The dynamic power consumption of the parallelized variant of NtruEncrypt, however, increases slower than that of the simple variant or Rabin's Scheme. This is observable as a slightly flatter slope at low frequencies. This effect is due to reduced switching activity in the Arithmetic Units compared to the rest of the circuit. With an increased number of AUs the difference becomes more noticeable. For ECC, the graph increases linearly, showing that even at 500 kHz the dynamic power consumption is already dominant.

Throughput The throughput that either architecture can achieve at a given clock frequency depends on the number of clock cycles for an encryption and the number of plaintext bits per block. In Rabin's Scheme the plaintext is up to 512 bits long.

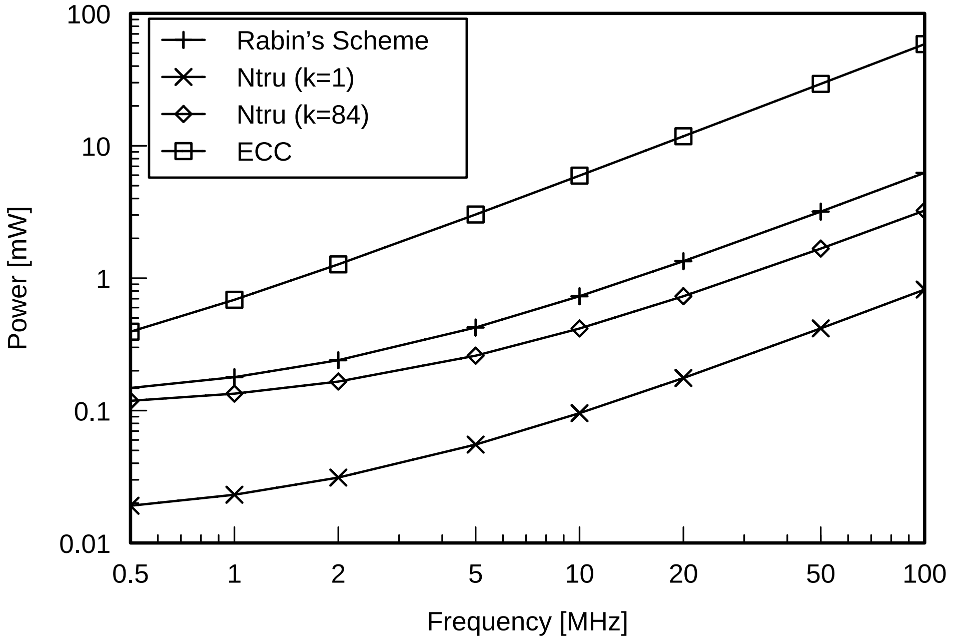

Figure 5.6: Power Consumption over a Range of Clock Frequencies

At a clock frequency of 500 kHz and an average of 1440 cycles per operation this translates into a maximum theoretical throughput of 177.8 kbits/s. Since NtruEncrypt uses N ternary coefficients we can determine its throughput in terms of kbits/s by first converting the capacity of the message polynomial $m(x)$ into bits. $N = 167$ ternary coefficients can hold information equivalent to $\lfloor N \log_2 3 \rfloor = 264$ bits. The entire encryption operation takes 29225 clock cycles for NtruEncrypt with a single AU, and 443 cycles with 84 AUs. Operating at the same clock frequency, the simple variant compares unfavorably to Rabin's Scheme at only 4.52 kbits/s throughput, almost 40 times less. The estimates for the highly parallelized variant, however, indicate a performance level of 304.85 kbits/s, nearly twice the throughput of Rabin's Scheme. The throughput of our ECC architecture is very slow as the necessary computations require many clock cycles and it has to complete two point multiplications for the ECMV algorithm. The plaintext for ECMV can be up to 200 bits long and it takes an average of 408,850 clock cycles to encrypt. This results in a throughput of only 0.24 kbit/s which is $1/740^{th}$ of Rabin's Scheme and $1/1270^{th}$ of the parallel NtruEncrypt.

Energy Efficiency For any cryptographic scheme there is a multitude of possible design choices by which power consumption can be traded off against performance and vice versa. Ultimately, however, we would like to know the amount of energy that is necessary for an elementary encryption operation, i.e. the cost of encrypting a bit of data at a certain level of security. The amount of energy for the entire operation is the product of average power consumption and the time it takes to complete that operation. Considering the amount of plaintext data that can be encrypted in one operation, we determine the amount of energy per bit encrypted as

$$E_{\text{bit}} = \frac{P_{\text{avg}} \cdot n_{\text{cycles}}}{f_{\text{clock}} \cdot l_{\text{op}}}$$

where l_{op} is the operand length in bits, i.e. 512 for Rabin's Scheme, 264 for NtruEncrypt and 200 for ECMV. As we have discussed earlier, Rabin's Scheme uses more power than NtruEncrypt with a single AU, but it also takes much fewer clock cycles to complete. We can make a similar observation by looking at the energy per bit metric. The amount of energy necessary to encrypt a single bit with NtruEncrypt is about five times higher than with Rabin's Scheme. The picture changes yet again when we consider NtruEncrypt's parallelized variant. Our estimates suggest that the amount of energy per bit drops by nearly factor 11 and is thus less than half the amount of Rabin's Scheme. The influence of parallelization on the amount of energy per bit can be seen in the graph in Figure 5.7. It turns out that $k = 84$ is not even the optimal value for energy efficiency. 56 parallel arithmetic units would give the optimal balance between power consumption and delay. The amount of energy per bit for ECMV is 1,612.5 nJ which is 380 times more than the 4.2 nJ for the simple version of NtruEncrypt. The smaller input size of ECMV, the higher power consumption and

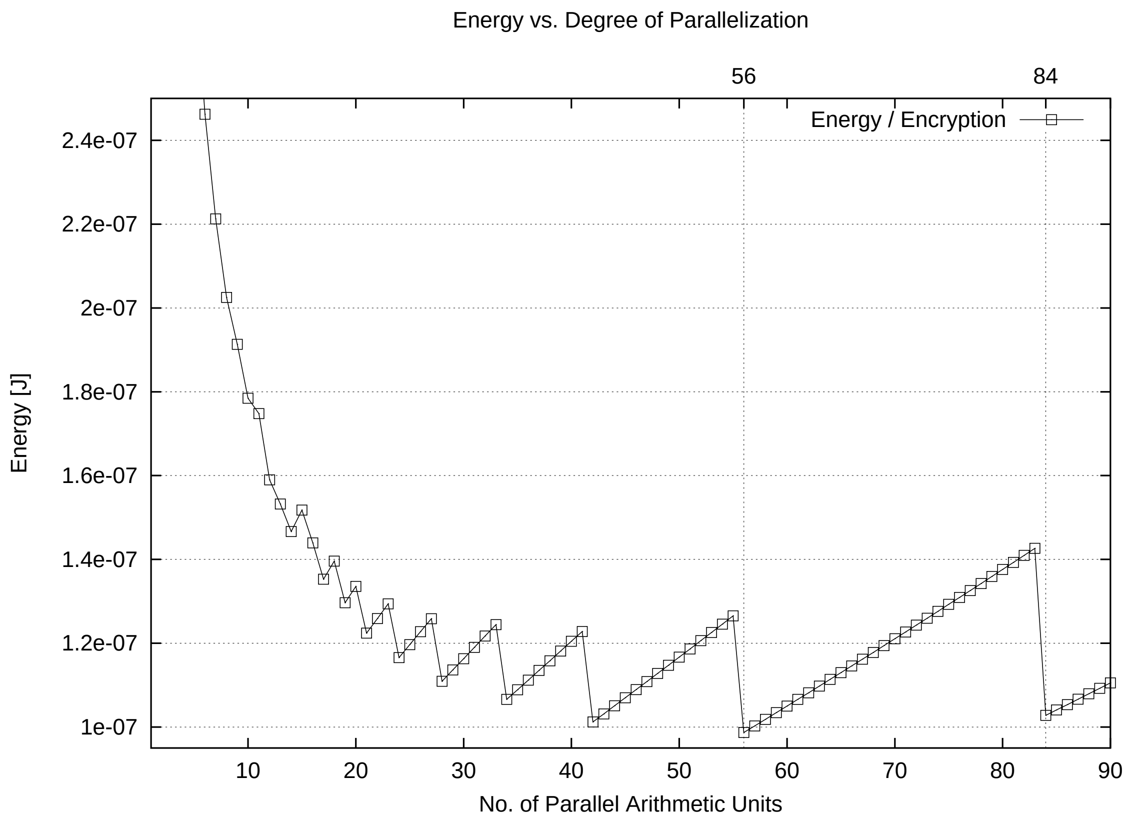

Figure 5.7: NtruEncrypt: Energy per Encryption Operation for $k = 1 \ldots 90$

the much larger delay all contribute to this high number.

To put our results into perspective, we compare them to estimates reported in [19] that were obtained from simulation of various public key algorithms on existing general purpose processor architectures. An implementation of the emerging scheme XTR on the ARC3 processor suggests an energy consumption of around 130 μJ at a security level that is comparable to RSA-1024 or 72 bits of equivalent security. Despite the difference in security levels, this is still between factor 100 and 1000 more energy than what our architectures require, proving the strength of customized application specific architectures.

5.5 Conclusions

We have demonstrated in this Chapter that it is possible to design public key encryption architectures with power consumption of less than 20 μW using the right selection of algorithms and associated parameters, optimization and low-power techniques. In spite of the common perception of public key cryptography, it is possible to achieve a level of power consumption low enough to allow its use even in self-powered sensor nodes. Our implementation is based on a regular ASIC standard cell library that is not specifically optimized for low-power. It is thus possible to achieve even better results than ours, although that is not the point we are trying to make here. The use of public key schemes facilitates much simpler security protocols than those currently in use with the sensor network community, and has a potential impact on a much wider range of applications. We explore this issue further in Chapter 7. RFIDs and contactless smart cards are further examples of ubiquitous computing applications requiring energy efficient cryptographic functions. So far public key cryptography has not even been considered for these devices due to its perceived complexity.

Our findings show further that schemes based on traditional modular arithmetic, such as Rabin's, does have a significant disadvantage compared to new and emerging schemes represented here by NtruEncrypt. The use of arithmetic in a polynomial ring allows for a very compact, yet scalable implementation in hardware. Additionally, NtruEncrypt's decryption operation—although not further considered in this Chapter—is based on the same arithmetic operation. This opens up the possibility for realization of two way key exchange protocols, while this is more difficult with Rabin's Scheme, due to its asymmetric properties of encryption and decryption.

The complexity of elliptic curve cryptography still seems to be prohibitively large for ultra low power applications. Further research into energy efficient cryptographic primitives is necessary, but our findings give us the confidence that public key cryptography in ubiquitous computing applications is possible and that it can be done efficiently using customized hardware architectures.

Chapter 6

Secret Key Functions

In this chapter we present a novel ultra-low power design of the popular Secure Hash Algorithm (SHA-1) and an energy efficient design of the ubiquitous Advanced Encryption Standard (AES). Both designs consume less than 30 μW of power and can therefore be used to provide the basic security services of encryption and authentication for ultra-low power devices. Furthermore, we analyze their energy consumption based on the TinySec protocol and come to the somewhat surprising result, that SHA-1 based authentication and encryption is more energy efficient than using AES for payload sizes of 17 bytes or larger. Parts of this work are published in [67].

6.1 Motivation

While working on hash functions (Chapter 4 and public key algorithms (Chapter 5 for ultra-low power implementations, we observed that at a frequency of 500 kHz leakage power becomes dominant. In order to conserve leakage power we have to reduce the circuit size. A common method to save hardware resources and provide privacy, integrity, and authentication is to use the same cryptographic algorithm for both functions, MAC computation and encryption. SPINS [108] for example, uses

RC5 [122] for encryption and in CBC-mode to build a secure MAC. TinySec [69] is cipher independent and was tested with RC5 and Skipjack [94, 95] for encryption and CBC-MAC. The authors of [69] are also considering the Advanced Encryption Standard [97].

Many research papers [108, 78, 81] analyze encryption algorithms for wireless sensor networks exclusively with reference to speed and code size while only a few [112] address the energy consumption of software based implementations. However, the ultra-low power applications we are envisioning, do not provide enough power for running cryptographic algorithms on general purpose microprocessors.

In this Chapter we are presenting hardware implementations of the advanced encryption standard (AES) and the Secure Hash Algorithm (SHA-1) [99] which are optimized for ultra-low power applications. We are then examining encryption and authentication functions based on AES and SHA-1. To our knowledge this is the first ultra-low power implementation of SHA-1 and the first publication describing the use of SHA-1 for ubiquitous computing on ultra-low power platforms. We are comparing SHA-1 and AES based encryption and authentication functions with respect to their footprint, speed, power and energy consumption. The only paper that compared SHA-1 to AES is [48], however the comparison is only concerned with throughput.

6.2 Introduction

We use AES and SHACAL-1 for encryption and AES in CBC-MAC mode and HMAC [9, 75] with SHA-1 for authentication. Figure 6.1 shows a top level view of the AES and SHA-1 based encryption and authentication functions.

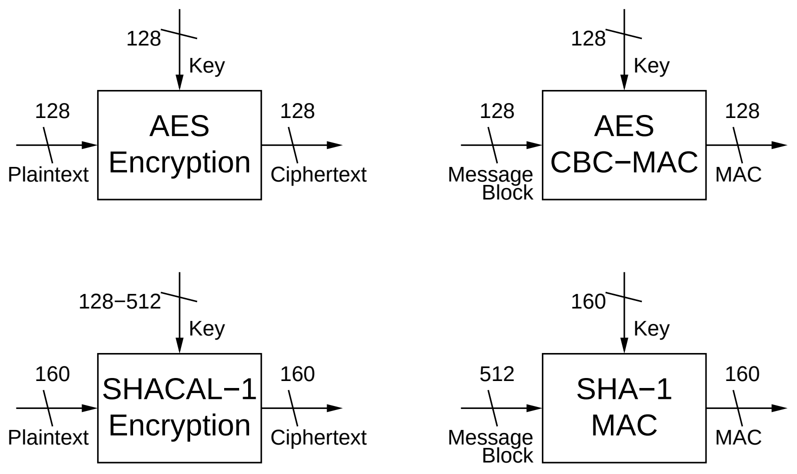

Figure 6.1: Encryption and MAC functions based on AES and SHA-1

6.2.1 AES

AES was selected by the National Institute of Standards and Technology (NIST) as Federal Information Processing Standard FIPS-197 [97] in 2001. Since then, many hardware implementations have been published. Most of them are optimized for speed and only a few are scalable [83, 45] from fast to small. The first ultra-low power implementation was reported in [38] followed by [37] by the same group, both papers analyze the power consumption but not the energy consumption of the circuits. AES is a block cipher with a fixed input size of 128 bits and a key length of either 128 bits, 192 bits, or 256 bits. For our ultra-low power implementation we chose 128 bits. AES applies the same round function ten times to its input, also called State, during encryption. The round function consists of four different transformations: SubBytes, ShiftRows, MixColumns, and AddRoundKey, each changing the State by applying linear, non linear and key dependent transformations.

6.2.2 SHA-1

SHA-1 is the most widely used secure hash function and was developed by the National Security Agency. Its security level is considered to be 2^{80}, i.e., 2^{80} operations have to be made on average to find another input such that the resulting hashes are equal, also called a collision. Recent attacks on SHA-1 [143] indicate that there might be a potential weakness but no collisions have been found yet. Many implementations of SHA-1 have been reported, most of them optimized for speed. To our knowledge, this is the first ultra-low power implementation of SHA-1. SHA-1 computes a 160-bit hash of messages up to 2^{64} bits in size. Each message needs to be preprocessed by padding the message, appending the message length and splitting it into blocks with a length of 512 bits each. Then the compression function processes each input block and computes intermediate hash values by iterating over simple functions 80 times.

6.2.3 Message Authentication Codes

SHA-1 can be used to build a message authentication code by introducing a secret 512-bit key K using the secret prefix method SHA-1($K||x$). Due to this concatenation SHA-1 will compute an intermediate hash value of K in the first iteration which can be precomputed. Hence, computation of a MAC requires $\lceil \text{length}(x)/512 \rceil$ operations. However, the secret prefix method is considered insecure [88] even though SHA-1 includes the message length in the hash and TinySec reveals only half of the hash result. We therefore suggest to use HMAC, which is formally described in [98] as

$$\text{HMAC}_k(x) = \text{SHA-1}((\overline{k} \oplus \mathit{opad}) \,||\, \text{SHA-1}((\overline{k} \oplus \mathit{ipad}) \,||\, x)).$$

The 160-bit key K is padded with 0's resulting in $\overline{k}$. The terms $\overline{k} \oplus \mathit{opad}$ and $\overline{k} \oplus \mathit{ipad}$ can be precomputed from the 512-bit constants *opad* and *ipad* and $\overline{k}$. Due to the concatenation $((\overline{k} \oplus \mathit{ipad}) \,||\, x)$ the intermediate hash value of $(\overline{k} \oplus \mathit{ipad})$ and $(\overline{k} \oplus \mathit{opad})$ can be precomputed as well. Hence, computation of a MAC requires

$\lceil \text{length}(x)/512 \rceil + 1$ operations of SHA-1.

AES can be used in CBC-MAC [135] mode (see Fig. 6.2) to compute authentication codes. This mode is similar to the *Cipher Block Chaining* mode [92, 93] in that

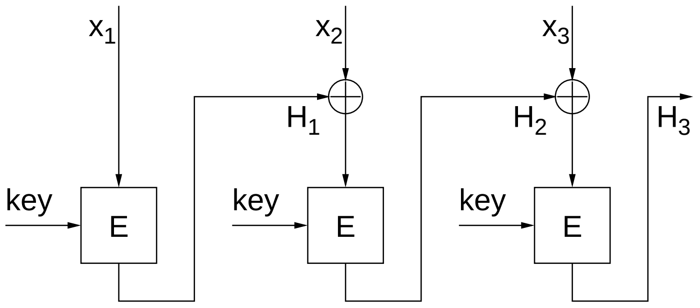

Figure 6.2: CBC-MAC – Generating a hash with a block cipher

the result from the previous encryption is XORed with the next plaintext block and encrypted again. The intermediary ciphertexts however are not used in CBC-MAC mode. The computation of a MAC requires $\lceil \text{length}(x)/128 \rceil$ operations. The level of security of AES in this mode is approximately 2^{64}, and not 2^{128} as one might expect, due to the birthday attack[1].

6.2.4 Encryption

To some extent, hash functions like SHA-1 can also be used to perform encryption. The best examples are SHACAL [50] and SHACAL-1 [51]. The security of SHACAL was analyzed in [52] and more recently in [125]. SHACAL defines how the compression function of SHA-1 can be used as a 160-bit block cipher with a 512-bit secret key. Shorter keys can be used by padding the key with zeroes but the minimum key size is 128 bits. AES is a block cipher so its usage for encryption is straight forward.

[1]If a sensor node produces one message authentication code per second, than it would produce only 2^{32} in 100 years.

6.3 SHA-1 Implementation

The top level block diagram of our SHA-1 implementation is shown in Figure 6.3. We assume that one 512-bit block of preprocessed data is stored in memory and available to our SHA-1 unit.

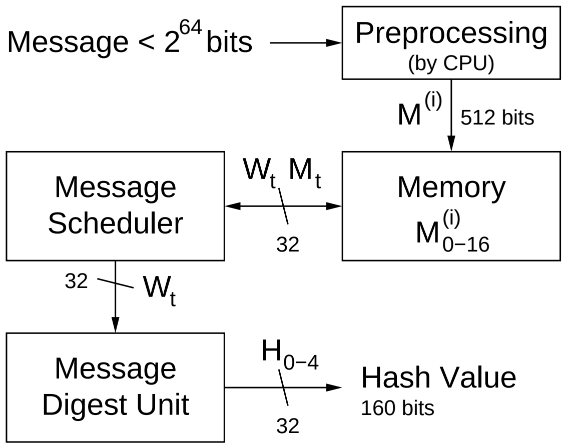

Figure 6.3: Top Level Block Diagram of our SHA-1 Implementation

Our SHA-1 implementation incorporates the *Message Scheduler* and the *Message Digest Unit* as well as a memory bus interface and the necessary control logic. The operation is broken down into three stages. The initial stage comprises the first 16 rounds. Here, the message scheduler reads the message block one $\mathrm{M_t}$ per round. The next stage is the computation stage which ends with the 80^{th} round. During both stages, the message scheduler computes $\mathrm{W_t}$ and forwards it to the message digest unit and also stores $\mathrm{W_t}$in the external memory. The message digest unit performs the message compression function. The final stage is needed to compute the final hash values from the intermediate hash.

6.3.1 Message Scheduler

During the computation stage the message scheduler has to compute a new $\mathrm{W_t}$ value in each round based on previously calculated $\mathrm{W_t}$'s. Most implementations in literature use a 16 stage 32-bit wide shift register for this purpose (512 flip-flops). For our ultra-low power implementation we re-use the memory that contains the message assuming that we can overwrite the existing contents. The message scheduler is able to interface with external memory and needs only one 32-bit register to store a temporary value during computation of the new $\mathrm{W_t}$. Figure 6.4 shows the block diagram of the message scheduler. The control logic, which handles the bus control signals, is not shown for simplicity. The message scheduler performs the equation

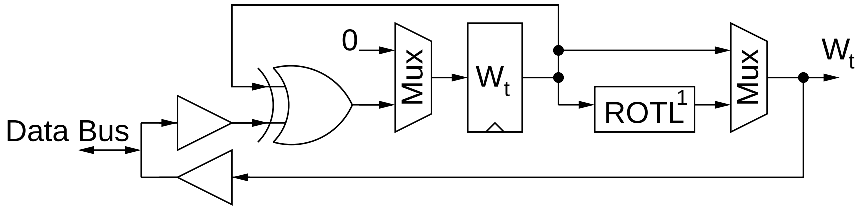

Figure 6.4: Block Diagram of the Message Scheduler

$$W_t = ROTL^1\left(W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16}\right)$$

where $\oplus$ denotes bitwise XOR. Four values have to be read from memory and the result written back to memory in each round. This takes 5 clock cycles in our serial design, therefore, each round of SHA-1 takes 5 clock cycles. The necessary address computation (not shown in Figure 6.4) is done using dedicated hard wired adders to provide +2, +8 and +13 addition modulo 16 for W_{t-14}, W_{t-8}, and W_{t-3} respectively.

6.3.2 Message Digest Unit

Figure 6.5 shows the functional block diagram of the message digest unit as described in the SHA-1 standard [99]. SHA-1 requires five 32-bit working variables (a, b, c, d,

e) to which new values are assigned in each round. It can easily be seen that four out of the five words are shifted in each round ($a \rightarrow b, \cdots, d \rightarrow e$) and only determining the new value for a requires computation. The rotation of $\mathrm{ROTL}^{30}(b) \rightarrow c$ can be accomplished by wiring. Therefore, we view the registers for the working variables as a 5 stage 32-bit wide shift register in our ultra-low power SHA-1 implementation.

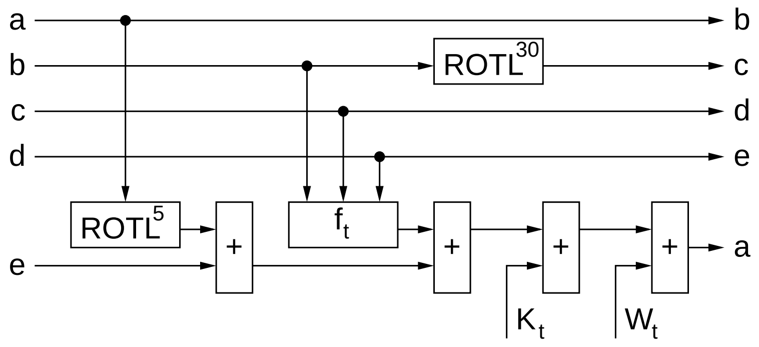

Figure 6.5: Functional Block Diagram of the Message Digest Unit

Round Function The round function computes a new value for a and shifts all working variables once per round. The computation for a is a five operand addition modulo 2^{32} where the operands depend on all input words, the round-dependent constant $\mathrm{K_t}$, and the current message word $\mathrm{W_t}$. In order to conserve area and therefore limit the leakage power, we use a single 32-bit adder to perform the four additions. Due to good scheduling of the operations, we can use the register for e also as temporary register for the additions. The four additions and the shift require 4 clock cycles per round which is below the need of the message scheduler with 5 clock cycles to compute the next $\mathrm{W_t}$. Figure 6.6 shows the block diagram of our implementation of the message digest unit including the round function and the intermediate hash value computation.

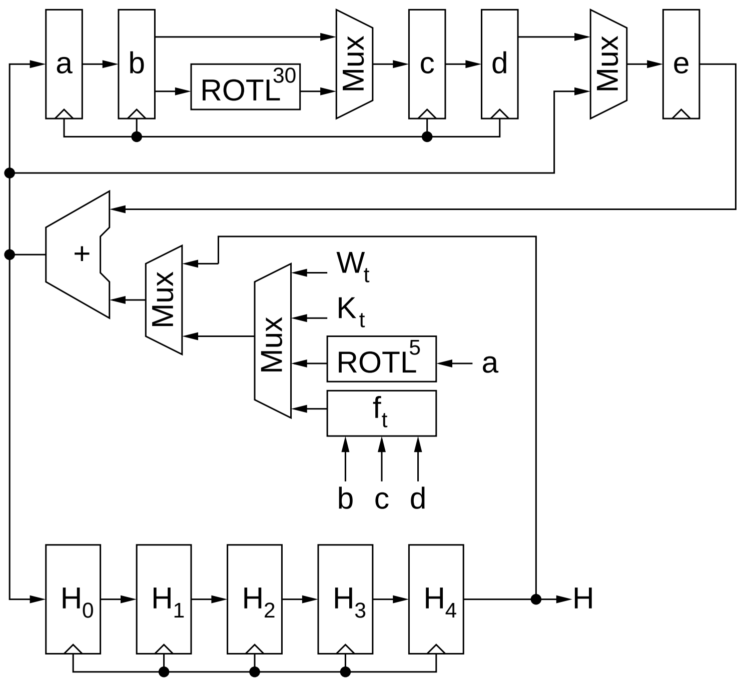

Figure 6.6: Proposed Hardware Architecture of the Message Digest Unit

Intermediate Hash Value Computation During the final stage, i.e., after the 80^{th} round, the values of the working variables have to be added to the digest of the previous message blocks, or specific initial values for the first message block. This can be done very efficiently without additional multiplexers or adders by arranging all intermediate hash value registers H_0, H_1, H_2, H_3, and H_4 in a 5 stage 32-bit wide shift register, similar to our design for the working variables. Shifting the hash value registers and the working variable registers at the same time and adding the current contents of e to H_4 at each step takes five clock cycles. This again fits into our scheme of 5 clock cycles per round, which leads to a total of 405 clock cycles for the message digest computation of one block.

6.4 AES Implementation

For our AES implementation we assume that a message block and the private key are stored in memory. The result of the AES computation gets written back to memory. Our 8 bit implementation is inspired by the one reported in [38], however, we restructured the datapath so that the registers get better utilized and the AES computation consumes less clock cycles. Fig. 6.7 shows the top level block diagram of our AES implementation. It consists of the Computation Unit, internal memory for key expansion and current state, one S-Box, a unit to compute the round constant Rcon, a control unit and a memory interface.

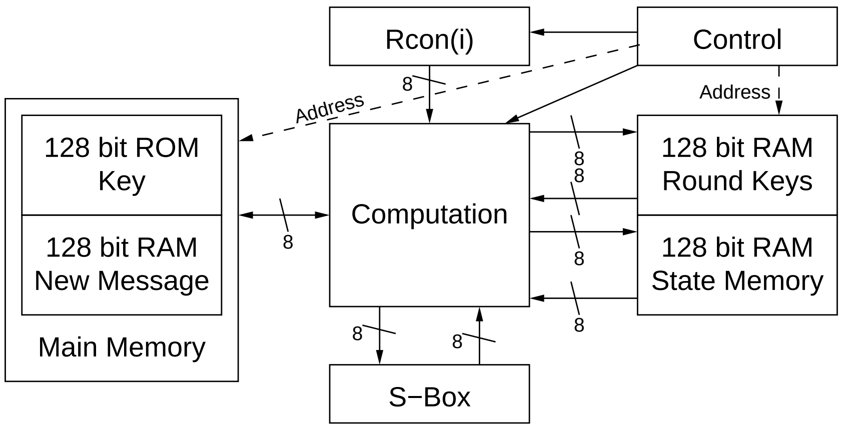

Figure 6.7: Top Level Block Diagram of our AES Implementation

In CBC-mode the hash of the previous message block gets XORed with the current message block. Therefore, we can not use the external memory to store the intermediate state as we could for our SHA-1 implementation. The same applies to storing the round keys.

6.4.1 Datapath

Each AES transformation and the key expansion load their operands in a specific order from the state memory or key memory respectively, and write them back. Some transformations require the storage of temporary results. We streamlined this process be grouping the AES transformations into four stages:

1. Initial AddRoundKey–SubBytes–ShiftRows
2. MixColumns
3. AddRoundKey–SubBytes–ShiftRows
4. FinalAddRoundKey

This grouping enables us to re-use registers and minimize the number of internal memory accesses. It allows us to use a pipelined architecture for stage 1 and 3 which reduces he number of clock cycles by 40 percent. This improvement comes at the cost of only one additional 8 bit register over the minimum possible number or 8 bit registers. Furthermore, the memory addressing scheme gets simplified. This is a tradeoff between low area and energy consumption.

The datapath of our implementation is shown in Fig. 6.8. It is characterized by the pipelined architecture for stage 1 and 3 as well as the register requirements for stage 2. We used five 8-bit registers, R_0, R_1, R_2, R_3, and R_4. The register R_0 is used exclusively for key storage and is needed to implement the *RotWord* operation. R_2, R_3, and R_4 are used for the state computation. R_1 is used for key computations except during the *MixColumns* operation where it gets reused for state computation. The boxes labeled *Keys* and *Data* are the register files for the Round Keys and the State Memory respectively.

Internal Memory The 128-bit state and the 128-bit round key are stored in internal memory. This memory is register based and makes extensive use of clock gating

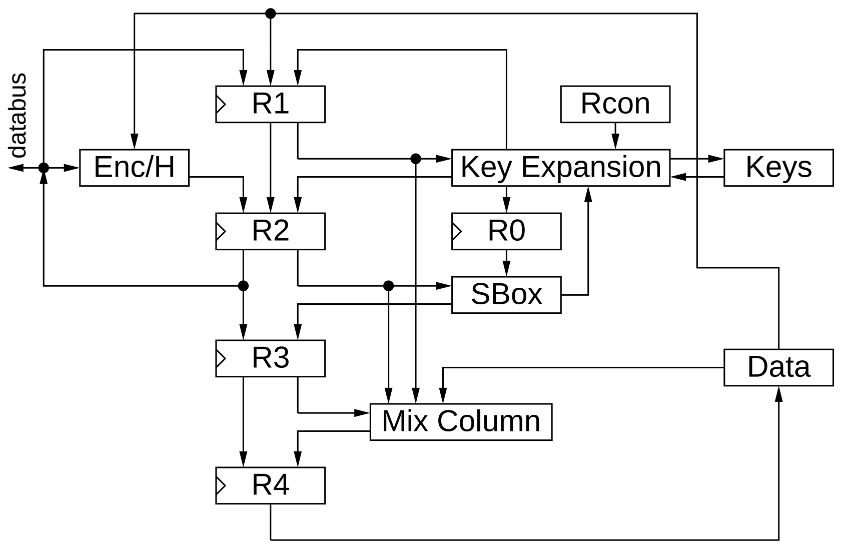

Figure 6.8: Block Diagram of our Implementation of the AES Datapath

to conserve power. The state memory has separate write and read addresses so that one value can be read while a value at another address can be written. All stages take advantage of this functionality due to the pipelined architecture. The key memory uses the same address for read and write.

6.4.2 Message Schedule

Initial AddRoundKey–SubBytes–ShiftRows During this first stage the 128-bit message block and the secret key are read from main memory. If used in CBC-mode, the message is XORed with the previous result. Then the the first AddRoundKey, SubBytes and ShiftRows operations are applied.

AddRoundKey–SubBytes–ShiftRows This stage is run nine times for AES. The round keys for the AddRoundKey operation are computed on the fly. This forces us to read data in column order from the state memory. The starting row is immaterial. The read order is $S_{r,0}$, $S_{r,1}$, $S_{r,2}$, $S_{r,3}, \ldots$ which translates to addresses for the state memory. As we merged the AddRoundKey operation and the ShiftRows operation the write order is predetermined. In order not to overwrite an element before its being read, we have to store four elements. Therefore the depth of our pipeline is four: R_1, R_2, R_3, and R_4.

Mix Columns Feldhofer et. al. [38] described a very efficient way for performing the *MixColumns* operation in an 8-bit architecture. It uses the minimum amount of registers needed for this operation. We used the same method, however we use an additional 8-bit register and are now able to reschedule the order of operations. The additional register (R_4) is available from the merging of *AddRoundKey* and *ShiftRows* operation. The new order of operations is shown in Equation 6.1.

$$\begin{array}{ccccccccc} S_{3,c} & \oplus & S_{2,c} & \oplus & (S_{1,c} \bullet \{03\}) & \oplus & (S_{0,c} \bullet \{02\}) & = & S'_{0,c} \\ S_{0,c} & \oplus & S_{3,c} & \oplus & (S_{2,c} \bullet \{03\}) & \oplus & (S_{1,c} \bullet \{02\}) & = & S'_{1,c} \\ S_{1,c} & \oplus & S_{0,c} & \oplus & (S_{3,c} \bullet \{03\}) & \oplus & (S_{2,c} \bullet \{02\}) & = & S'_{2,c} \\ S_{2,c} & \oplus & S_{1,c} & \oplus & (S_{0,c} \bullet \{03\}) & \oplus & (S_{3,c} \bullet \{02\}) & = & S'_{3,c} \end{array} \tag{6.1}$$

This order of operations results in the read order: $S_{0,c}$, $S_{1,c}$, $S_{2,c}$, $S_{3,c}$, $S_{0,c}$, $S_{1,c}$, $S_{2,c}, \ldots$. This order of addresses is now very similar to the one needed for the *AddRoundKey* function with row and column addresses swapped. This simplifies the address computation in the control logic.

Final AddRoundKey In this stage we perform the final round key computation and AddRoundKey operation. Then the result is written back to memory. Hence, the cipher can be used for encryption in CBC mode, as well as hash function in CBC-MAC mode.

6.5 Analysis and Comparison

All our designs were described in VHDL and verified by simulation with ModelSim and test vectors from the respective standards [97, 99]. We synthesized the VHDL code using Synopsys and used ModelSim for further verification and switching activity analysis. Our target library is a 0.13μm, $V_{DD} = 1.2$ V ASIC library from TSMC, which is characterized for power. The final results for power, area, and delay were reported by Synopsys Power Compiler at the gate level. We would like to emphasize that our contribution is on the algorithmic and architectural level. Implementing our designs using an ultra-low power ASIC library or a full custom chip design will enable higher energy and power savings. We implemented SHA-1 once with a carry look ahead adder (CLA) and once with a carry propagate adder (CPA). The results for both SHA-1 implementations and for AES are shown in Table 6.1. All designs consume a similar amount of area and power. However, the critical path delay in SHA-1 (CPA) is more than twice as long as for AES. The critical path in AES includes the S-Box, which is the most complex part of the circuit. The delay of SHA-1 (CPA) is caused by a 32-bit CPA, also called ripple carry adder. We implemented a carry look ahead adder (CLA) for SHA-1 in order to reduce the critical path delay. However, our SHA-1 (CLA) implementation exhibits both, higher leakage power due to the increase in area caused by the additional logic for the carry look ahead, and larger dynamic power consumption. Therefore, we consider only SHA-1 (CPA) for the following comparisons. The power consumption of the SHA-1 (CPA) and AES designs is computed for a 500 kHz clock, which is far below their maximum frequency. The total power consumption of SHA-1 is about 10 % higher than that of AES. Within 534 clock cycles AES can encrypt 128 bits of plaintext. SHA-1 needs 405 clock cycles to compute the hash of 512 bits of data.

Feldhofer et.al. presented two related AES designs in [38] and [37] consuming 26.9 μW and 4.5 μW respectively with a 100kHz clock. These numbers are difficult to

Table 6.1: Results for SHA-1 and AES

	SHA-1 (CLA)	**SHA-1 (CPA)**	**AES**
Maximum critical delay	3.17 ns	5.72 ns	2.19 ns
Clock cycles for one operation	405	405	534
Area (NAND equiv.)	4362	4276	4070
Static Power	23.55 μW	23.00 μW	20.23 μW
Dynamic Power (at 500 kHz)	3.95 μW	3.74 μW	3.60 μW
Total Power (at 500 kHz)	27.49 μW	26.73 μW	23.83 μW

compare with our design as the results for power consumption are highly technology dependent. The encryption only design in [38] consumes an area of 3595 NAND equiv. and needs 1016 clock cycles. The design in [37] needs 3400 NAND equiv. and 1032 clock cycles. Both designs do not support CBC mode which requires extra hardware. It can easily be seen that our implementation uses 20% more hardware resources than their smallest design while using 48% less clock cycles, i.e. it is almost twice as fast. The slight increase in hardware resources leads to large decrease in computation time which reduces the energy consumption while still being an ultra-low power circuit. For a fair comparison of AES and SHA-1, we used the same implementation and optimization techniques with the same ASIC library.

In order to explore the energy consumption of our AES and SHA-1 implementations we focus on the TinySec [69] protocol. Table 6.2 shows the results assuming the TinySec packet format and a payload of 29 bytes.

Table 6.2: Energy Results for SHA-1 and AES (29 bytes/packet, 500 kHz)

	MAC		**Encryption**		**Encryption & MAC**	
	AES	**SHA-1**	**AES**	**SHA-1**	**AES**	**SHA-1**
Energy [nJ]	76.42	43.32	50.95	43.32	127.36	86.64
Power [μW]	23.85	26.74	23.85	26.74	23.85	26.74
Time [ms]	3.20	1.62	2.14	1.62	5.34	3.24
Energy/bit [nJ]	0.33	0.19	0.22	0.19	0.55	0.37

6.5.1 Message Authentication Codes

Due to the lossy nature of low power wireless transmission it is not feasible to compute a single MAC for multiple packets. Therefore, SPINS [108] computes a MAC for each packet using RC5 [122] in CBC-MAC mode and appends it to the original packet. TinySec defines a packet format for authenticated messages (TinySec-Auth) that can carry up to 29 Bytes of payload. The MAC is computed over the payload and the packet header which is four bytes long. Table 6.2 shows that using AES to compute the MAC over 29+4 bytes consumes $76.42\,\mu$J and SHA-1 consumes $43.32\,\mu$J. Even though SHA-1 consumes 10% more power than AES, the running time of AES is larger by a factor of two, leading to the higher energy consumption. Fig. 6.9 shows the energy consumption for MAC computation over different payload sizes, each time assuming a four byte overhead. Until the payload reaches 29 bytes AES consumes less or almost equally as much energy as SHA-1. For payloads of 29 bytes or larger AES has to run more than twice while for SHA-1 two iteration are sufficient, due to its longer input size.

6.5.2 Encryption

Even though TinySec does not specify an encryption only format we still consider it for comparison purposes. We assume that only the payload has to be encrypted and the packet header is transmitted in the clear. Table 6.2 shows that the difference in Energy consumption between SHA-1 and AES are less dramatic for encryption than for MAC computation. Fig. 6.10 shows that SHA-1 follows AES closely. This comes from the fact that the input size of AES is 128 bits and of SHA-1 in encryption mode (SHACAL-1) is 160 bits.

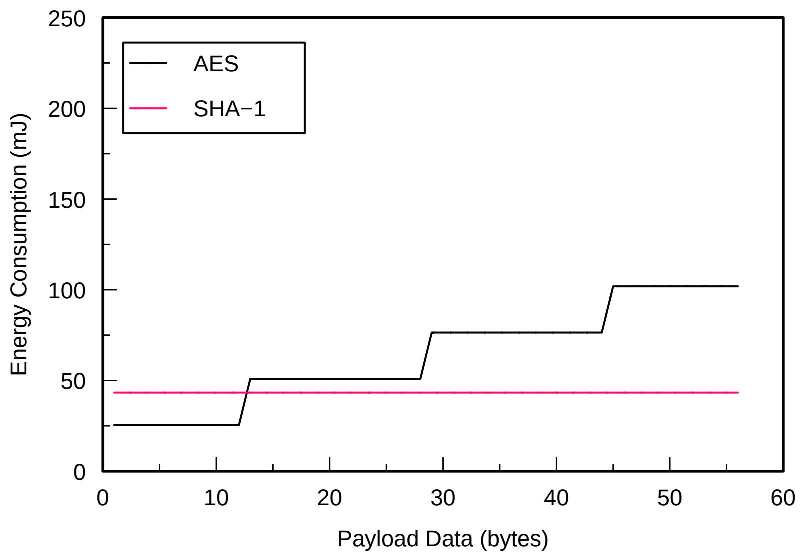

Figure 6.9: Energy Consumption of MAC Computation with AES and SHA-1 Depending on Payload Size

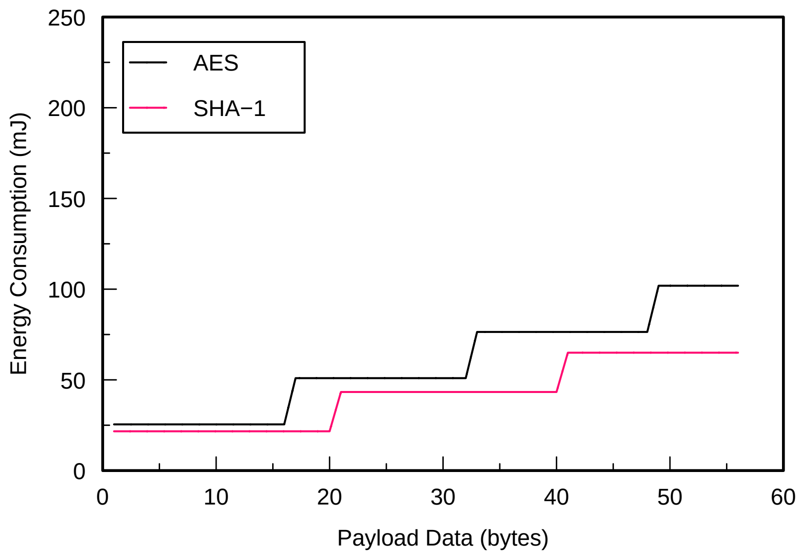

Figure 6.10: Energy Consumption of Encryption with AES and SHA-1 Depending on Payload Size

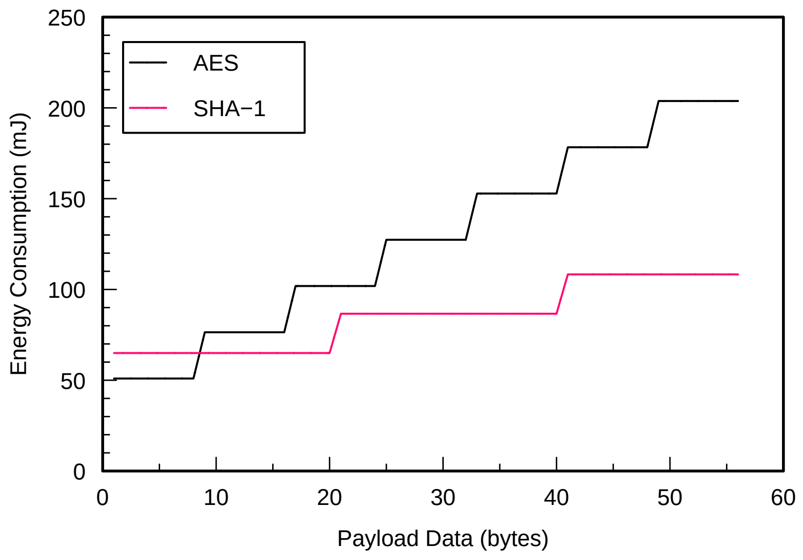

Figure 6.11: Energy Consumption of Encryption and MAC Computation with AES and SHA-1 Depending on Payload Size

6.5.3 Authentication and Encryption

The packet format for Authentication and Encryption (TinySec-AE) specifies a payload of upto 29 bytes and a packet header of eight bytes length. Only the payload has to be encrypted but the MAC is computed over the payload and the message header. Assuming a 29-byte payload, AES consumes almost 1/3 more energy than SHA-1 (see Table 6.2. For larger payloads the SHA-1 consumes significantly less power (see Fig. 6.11).

6.6 Conclusion

In this Chapter we presented a novel ultra-low power implementation of SHA-1 and an ultra-low power and low energy AES design. Both circuits consume less than 30 μW of power and could therefore be powered by scavenger circuits. We analyzed the energy consumption of SHA-1 and AES based encryption and message authentication functions. The result of our analysis is that SHA-1 and AES seem to be equally well suited for ultra-low power applications if the payload size is below 17 bytes. For payloads of 17 bytes or above SHA-1 needs significantly fewer iterations than AES and therefore a shorter running time which conserves energy. We want to emphasize that the power consumption of both algorithms is about the same.

Chapter 7

Security Protocols

Security in wireless sensor networks was until recently provided exclusively through symmetric key cryptography. In this Chapter we show that the special purpose ultra-low power hardware implementations of public key algorithms from Chapter 5 can be used on sensor nodes to enable services like broadcast authentication and to reduce the protocol overhead of other services. This translates into less packet transmissions and hence, power savings. We provide an in-depth comparison of three popular public key implementations and describe how four fundamental security services benefit from PKC.

7.1 Introduction

Most publications, seemed to preclude that public key cryptography (PKC) is not feasible on severely resource constrained sensor nodes, and therefore revert to emulation of asymmetry using symmetric key techniques in security protocols such as μTESLA [108]. Most, if not all, implement cryptographic primitives in software on general purpose micro-controllers. In this Chapter we present the two most popular cryptographic protocols for WSN and show that PKC tremendously simplifies the implementation of many typical security services and additionally reduces transmission power due to less protocol overhead. Moreover, the capture of a single node would not compromise the entire network, since no globally shared secrets are stored on it. Our approach to overcome the difficulty in implementing PKC in sensor nodes is based on providing a custom-designed low-power co-processor that can be embedded in the node and that handles all of the compute-intensive tasks.

In section 7.3 we identify four fundamental security services that would particularly benefit from PKC. Based on the analysis of the three low-complexity PKC architectures presented in Chapter 5 we estimate the overall power and bandwidth requirements of encryption and signature primitives in Section 7.4. The analysis of the results in Section 7.4.2 with respect to the previously mentioned security services can serve protocol designers as a guideline for incorporating public key-based services into their WSN protocols.

7.2 Popular Protocols

7.2.1 SPINS with SNEP and μTESLA

SPINS is a security protocol optimized for sensor networks and was presented in [108]. It consists of two parts: SNEP and μTESLA which run ontop of TinyOS [80][55]. TinyOS is a small, event driven operating systems for sensor nodes with 8-bit low end microprocessors.

SNEP SNEP stands for Secure Network Encryption Protocol and is used to provide confidentiality through encryption and authentication as well as integrity through a message authentication code. It uses a counter to provide the service called freshness which allows detection of replayed messages. Encryption is provided through the block cipher RC5 in counter mode and MAC computation through the same block cipher in CBC-MAC mode (see Section 6.2.3).

μTESLA is the "micro" version of the Timed Efficient Stream Loss-tolerant Authentication (TESLA) scheme proposed in [107]. It emulates asymmetry through a delayed disclosure of symmetric keys and serves as the broadcast authentication service of SNEP and TinySec. Unlike TESLA, which authenticates the initial packet using a digital signature, μTESLA relies solely on delayed disclosure. While Carman et al. acknowledge in [19] that symmetric key techniques are attractive due to their energy efficiency, they also conclude that all symmetric key based key exchange protocols analyzed by them exhibit limitations in their flexibility.

μTESLA and other schemes based on delayed key disclosure suffer from a serious denial of service attack [117]. The μTESLA specification states that the base station sends broadcast packets with a message authentication code for which the key has not been disclosed. Each node is required to store all messages received during one time interval for which the key has not been disclosed and forward them to other nodes in the tree. In the subsequent time interval, the base station is disclosing the old key, and the nodes can verify all messages in their buffer. The requirement that each node has to buffer and relay all messages for which it has not received a valid key can easily be exploited by an attacker. He can flood the network with arbitrary messages, claiming that they belong to the current time interval. Only in the next time interval the nodes are able to verify that these messages are not authentic. This can lead to buffer overflows in the nodes and to battery exhaustion as all messages have to be forwarded to other nodes.

The use of public key cryptography would eliminate the need for complicated protocols and at the same time would also increase the security of the entire system, since only the public key of the base station would have to be embedded into the nodes.

7.2.2 TinySec

TinySec [69] replaces SNEP. Therefore it provides similar services, namely authentication, integrity, confidentiality and replay protection. Unlike SNEP, it does not use counters. It uses CBC mode with ciphertext stealing [129] for encryption and CBC-MAC for authentication. In order to make the CBC-MAC secure for variably sized messages TinySec XORs the encryption of the message length with the first plaintext block [10]. TinySec defines two packet formats: TinySec-Auth (section 6.5.1) for authenticated messaged and TinySec-AE (section 6.5.3) for authenticated and encrypted messages. Furthermore, TinySec is integrated into TinyOS.

7.3 Security Services

In this section we state our assumptions regarding the structure of the WSN and define an exemplary subset of four security services that would particularly benefit from the use of PKC.

Sensor networks typically consist of a number of tiny nodes communicating with a base station [108]. The base station collects the data from the sensors and communicates with the outside world. The sensor nodes only have limited power and can therefore communicate directly only with nodes in close proximity. They establish a routing tree with the base station at its root. The base station is assumed to have sufficient power for all computations and communications with the nodes and the outside world.

Broadcast Authentication In this scenario, a base station would, for example, broadcast a set of commands, to all sensor nodes at once. Each sensor node would need to verify that this message originated from the trusted base station and not from an adversary. This scenario is a typical application for PKC. All nodes would need to have the base station's public key embedded. Data recovered from captured nodes would not be helpful to the adversary in forging messages. Previously published schemes either require large amounts of data to be sent or a complicated symmetric key release scheme [108] which is vulnerable to a denial of service attack.

Data Encryption Data encryption using PKC is much more expensive than using secret key cryptography. However, in certain cases where no secret key is established, it can be useful. One case is node-to-node key distribution described below. Another scenario could be, that a sensor node has to send some data all the way to the base station. The node just needs the base station's public key for this operation.

Node-to-Node Key Distribution Another typical PKC application is key distribution and key agreement. Key agreement refers to a protocol where two parties jointly establish a key, whereas key distribution is defined as a protocol where one party securely transmits a key to another party. Here we are assuming that each node knows the public key of its neighbors. The private key could be distributed during the routing setup phase or by querying the base station. If two nodes want to establish a session key a node simply encrypts it using its neighbors public key and sends it. Unlike other schemes, the base station does not need to get involved, which saves transmission power.

Addition of new Nodes New legitimated nodes might need to be added to a WSN at any point in time. These nodes must be included into the security scheme of the WSN. Again, PKC offers an elegant solution. Each sensor node has its own public/private key pair and additionally the base stations public key. The public keys of the new nodes can be sent via the outside communications link to the base station. Now the base station can trust the new nodes and vice versa. The base station can encrypt a secret session key with the node's public key and send it, or the node can announce its arrival in the network by sending a signed message to the base station. There is no need for additional bootstrapping information.

7.4 Feasibility Study

In Section 7.3 we identified four security services that would benefit from an efficient ultra-low power PKC implementation. Here we identify which PKC function is needed for each of these services. *Broadcast Authentication* uses signature verification on the node using the base station's public key. In order to provide the *Data Encryption* service for sending data to the base station, the node has to encrypt data, again using the base station's public key. *Node-to-Node Key Distribution* requires encryption as well as decryption. *Addition of New Nodes* is based on a node having a private key. The node would either sign a message and send it to the base station, or decrypt a message received from the base station. Table 7.1 provides an overview of these functions for the three PKC systems that we consider.

7.4.1 Public Key Schemes

Now we are showing which PKC can support the above mentioned services and provide estimates on the power consumption and throughput.

Rabin's Scheme as defined in [115] can be used for all four methods. Chapter 5.2.2 shows how it can be used for data encryption. This is the same function as signature verification. Data decryption, as well as signature generation, requires solving the equation $\mathrm{D}_n(x) \equiv \sqrt{y} \mod n$. If we set $p \equiv q \equiv 3 \mod 4$ then the square root can be computed elegantly using Euler's criterion and Garner's algorithm as follows. We compute the solution for $y^{(p+1)/4} = c_1 \mod p$ and $y^{(q+1)/4} = c_2 \mod q$ separately. Using a slightly modified Garner's algorithm we compute the result x as $x = \pm c_1 +$

$[(\pm c_2 \pm c_1) \cdot p(p^{-1} \mod q)] \mod n$. The exponents $(p+1)/4$ and $(q+1)/4$ as well as the factor $p(p^{-1} \mod q)$ can be precomputed and hard wired, just like the keys. A decryption takes two exponentiations with an exponent of at most 255 bits and one multiplication $\mod n$. We ignore the cost of the additions as it is negligible compared to the multiplication cost. The decryption will need 762 multiplications on average with 255-bit coefficients and one 512-bit multiplication. If we employ the same circuit as we use for encryption (which is not an optimal solution) then one decryption would take on average 544,753 clock cycles. That means a single decryption or signing would take 1.09 seconds. This new circuit would require more storage for c_1 and c_2, additional multiplexers for the precomputed constants and a more complex control logic. Conservative estimates result in a total power consumption of 191.5μW.

NtruEncrypt and NtruSign Our basic NtruEncrypt encryption primitive provides us with representative data from which we can extrapolate power and energy requirements for the decryption, signature generation and signature verification procedure. In our original architecture we fixed the public key as a constant in a very compact look-up table. For our estimates of the other primitives we therefore add an overhead of around 40 μW of static power to our simulation results that caters for the additional storage requirements. We base our estimates further on the number of cyclic convolutions that are required by the respective primitive, since that is the central arithmetic operation in all Ntru schemes.

Based on [59, 58] we found the number of convolution operations to be 1, 2, 4 and 1 for encryption, decryption, signature generation and verification, respectively. Convolution is by far the most complex operation in NtruEncrypt and NtruSign, so it is safe to assume that time and energy are proportional to the number of convolutions. The figures for energy consumption are the products of the time and power estimates.

ECMV and ECDSA The elliptic curve based encryption and signature algorithms we selected are all based upon scalar point multiplications. According to the description of these algorithms in several standards and publications, the encryption and signature verification primitives of ECMV and ECDSA each require two scalar point multiplications, while for decryption and signature generation a single scalar point multiplication is sufficient. We base our time estimates on these findings, coupled with the performance figures for our baseline ECC architecture.

7.4.2 Comparison

Table 7.1 compares the PKC functions with regards to speed, power, energy and message length. The transmission power for the messages is not included as we did not consider a particular transmission system. However, the length of the ciphertext and signature can be used for an estimate for a particular transmitter. For encryption and decryption the ratio of payload length vs. ciphertext length is important. Signatures are transmitted in addition to the original message.

Typical packet sizes on WSN are 30 bytes [108] and 56 bytes. Due to its asymmetry, Rabin's scheme is particularly suitable if only encryption and signature verification are performed on the node. Otherwise it is comparable to ECC. Ntru has the smallest average power consumption, but the largest message size of 5 packets. In environments where transmission power is not the most dominant part, Ntru has an advantage. ECC has a small message expansion for encryption and a high power consumption but requires the smallest number of packets. Also, the message content (key material) rarely exceeds 200 bits. On most WSN nodes, transmitting a single bit costs as much power as executing 1000 instructions. Small message sizes and low overhead is of utmost importance which is a feature of ECC.

Broadcast Authentication can benefit greatly from using a PKC. With ECC only one additional packet needs to be sent to authenticate a message from the base station.

Protocols like μTESLA [108] require a complicated delayed key disclosure scheme which is vulnerable to denial of service attacks, requires constant key updates, the nodes have to store keys, and be time synchronized. Bootstrapping a new node becomes especially difficult. *Node-to-Node Key Distribution* can now be done with only two (ECC) or three (Rabin) packets between the nodes. Involving the base station in this key setup becomes especially expensive if the communicating nodes are many hops away. The scheme presented in [108] requires at least four messages, three of which involve the base station. The details of when *Data Encryption* is advantageous and how the *Addition of new Nodes* is handled is dependent on the specific protocol. However, our results indicate, that only very few packets are necessary with PKC.

Table 7.1: Comparison of PKC Functions (Packets of 30 bytes)

Encryption/Decryption	Rabin	NtruEncrypt	NtruEncrypt parallel	ECMV
- Message Payload	< 512 bits	< 265 bits	< 265 bits	< 200 bits
- Ciphertext	512 bits (3)	1,169 bits (5)	1,169 bits (5)	400 bits (2)
	Encryption			
Time per Message	2.88 ms	58.45 ms	0.87 ms	817.7 ms
Avg. Power	148.18 μW	19.13 μW	118.7 μW	394.4 μW
Energy per Message	426.76 nJ	1,118.15 nJ	102.79 nJ	322.5 μJ
	Decryption			
Time per Message	1.089 s	116.9 ms	1.732 ms	411.54 ms
Avg. Power	191.5 μW	58.73 μW	158.3 μW	394.4 μW
Energy per Message	208.64 μJ	6,865.54 nJ	274.18 nJ	162.31 μJ

Sign / Verify	Rabin	NtruSign	NtruSign parallel	ECDSA
- Signature Length	512 bits (3)	1,169 bits (5)	1,169 bits (5)	200 bits (1)
	Sign			
Time per Message	1.089 s	233.8 ms	3.464 ms	410.45 ms
Avg. Power	191.5 μW	58.73 μW	158.3 μW	394.4 μW
Energy per Message	208.64 μJ	13.73 μJ	548.35 nJ	161.88 μJ
	Verify			
Time per Message	2.88 ms	58.45 ms	0.87 ms	822.5 ms
Avg. Power	148.18 μW	19.13 μW	118.7 μW	394.4 μW
Energy per Message	426.76 nJ	1,118.15 nJ	102.79 nJ	324.39 μJ

Chapter 8

Conclusion

This Chapter summarizes the results of the research presented in this dissertation and suggests directions for future work in this area.

8.1 Summary and Conclusion

Wireless Sensor Nodes (WSN) and Radio Frequency Identification Devices (RFIDs) belong to a new set of ultra-low power applications which make computing ubiquitous. WSN and RFIDs are quickly becoming a vital part of our infrastructure [29]. Security is a critical factor for these ultra-low power devices due to their impact on privacy, trust and control. Both technologies impose severe power and area constraints on the underlying hardware devices. Traditional cryptographic algorithms are considered too bulky, complex and power hungry for these devices. The goal our research was to develop a suite of cryptographic functions for authentication, encryption, and integrity that is specifically fashioned to the needs of ultra-low power devices. This includes public key cryptography, secret key cryptography, message authentication codes and secure hash functions.

Developing hardware implementations of cryptographic algorithms for ultra-low power devices is not as straightforward as compiling existing VHDL code for an low-power ASIC library. We carefully selected several algorithms that seemed promising for a ultra-low power implementation. We made extensive use of power saving techniques on the architectural, logic, and system level (e.g. clock gating, operand isolation) when we implemented the algorithms. In most cases the speed of the algorithm is not as important as the power consumption. At the low clock frequencies of these devices leakage power is dominant. Therefore, we minimized the power consumption by minimizing the circuit size. Furthermore, we have to look at security protocols and evaluate if a specific algorithm might save on the overall transmissions as transmission power is very expensive.

Universal Hash Functions for Ultra-Low Power Devices Protecting the integrity of data is of utmost importance for many application scenarios of ubiquitous computing. In many cases the data transmitted between sensor motes or an RFID tag and the reader is not confidential but its authenticity and integrity are very important. Universal hash functions, first introduced by Carter and Wegman provide a unique solution to the aforementioned security problems. A universal hash function family can be used to build an unconditionally secure MAC. When we implemented a universal hash function family (NH) we identified several possible hardware optimizations. Some of them involved removing registers, multiplexers and several gates. The result was a new hash function family (WH) which we could prove to provide better security characteristics than NH. This approach of developing new algorithms seems very promising. Furthermore we investigated techniques like multi-hashing, and the Toeplitz approach to reduce the energy consumption through leakage power even more. Our work is described in [153] which describes the development of WH and in [68] which describes how we achieve energy scalability for our universal hash function family.

Public-Key Cryptography for Ultra-Low Power Devices It was widely believed that public key cryptography is not feasible on sensor nodes. Many elaborate wireless security protocols have been designed to emulate public key features using only secret key functions. However, recent results [117] have shown that this can lead to vulnerabilities. Our work on public-key cryptography for ultra-low power devices published in [42] was to our knowledge the first that addressed this issue and is now heavily cited. We designed proof-of-concept hardware implementations of three distinct fundamental functions covering the three major areas of public key cryptography. We followed up on this work by showing how the use of these functions can lead to very simple and energy efficient security protocols, saving expensive transmission power [41].

Secret Key Cryptography To complete our suit of cryptographic functions for ultra-low power devices we analyzed and implemented AES and SHA-1 for authentication and encryption. SHA-1 has a larger input size than AES so one would expect that in order to encrypt a large block of data SHA-1 needs less iterations than AES. The same applies for computing the message authentication code of the same block of data. Our surprising result was, that this property manifests itself already for small inputs of only 17 bytes, i.e., our SHA-1 design is more energy efficient than our AES design for any data block larger than 17 bytes.

In summary, we presented several ultra-low power proof-of-concept implementations of cryptographic algorithms covering all basic cryptographic services. We see our work as a foundation for future research.

8.2 Recommendations for Future Research

In this dissertation a suite of cryptographic functions for ultra-low power applications was developed. During this process ideas have surfaced that expand the scope of the original goals. This section provides the reader with an overview of these ideas which represent possible areas in which further work could be pursued.

Technological Level Optimizations In this dissertation, we made extensive use of power saving techniques at the architectural, logic, and system level. Power optimizations of the technological level can lead to further power savings. It would also enable us to investigate the effects of different technologies for memory implementations that have the potential of consuming less power than flip-flops [114]. Ultra-low power memory can offer interesting tradeoffs for serialization and precomputation. Another important point would be to invesigate how our results will scale to future CMOS technologies.

Investigating other Algorithms and Applications So far, we analyzed the most important cryptographic algorithms. In the near future this can be expanded to other algorithms like Hyper Elliptic Curves, XTR, Kasumi, etc. This topic can also be expanded applications other than WSN and RFID which might have different power, area, or transmission requirements.

Making Ultra-Low Power Cryptographic Devices Tamper Proof This topic covers the physical security of the devices as well as side channel attacks. Many applications of wireless sensor nodes and RFID tags makes them vulnerable to attacks. The physical implementation of cryptographic algorithms can leak information about secret data to an attacker through side channels e.g., fluctuation in power consumption, electro magnetic radiation, etc. Techniques to thwart these attacks are currently being developed. One area for research could be to study how these techniques can be applied for ultra-low power implementations without exceeding power and area limitations.

Ultra-Low Power True Random Number Generators The security of almost all cryptographic systems depends on the randomness, unpredictability and secrecy of the key. Many cryptographic protocols require random numbers also for purposes other than the key. Therefore, a true random number generator (TRNG) must meet stringent requirements. Designing TRNGs is the subject of current research. Most published implementations have large hardware requirements. It would therefore be very interesting to study how a TRNG can be built within the constraints of ultra-low power applications.

Application to Secure Wireless Networks The research presented in this dissertation resulted in proof-of-concept implementations for ultra-low power cryptography. As a next step we could deploy the cryptographic hardware implementations in actual WSN motes or on RFID tags. Through this, we would obtain clearer definitions and constraints for further research.

Bibliography

[1] TI celebrates 10 year aniversary of RFID. http://www.ti.com/rfid/docs/manuals/RFIDNews/Tiris_NL20.pdf, 2000. RFID News, Issue 20, Texas Instruments.

[2] Wal-Mart details RFID requirement. RFID Journal, Nov 2003. http://www.rfidjournal.com/article/articleprint/642/-1/1/.

[3] Katherine Albrecht and Liz McIntyre. *Spychips : How Major Corporations and Government Plan to Track Your Every Move with RFID*. Nelson Current, 2005.

[4] M. Alidina, J. Monteiro, S. Devadas, A. Ghosh, and M. Papaefthymiou. Precomputation-based sequential logic optimization for low power. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2(4):426–436, Dec 1994.

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

[6] R. Anderson, E. Biham, and L. Knudsen. Serpent: A proposal for the advanced encryption standard. In *First Advanced Encryption Standard (AES) Conference*, Ventura, California, USA, 1998.

[7] A. Arora, P. Dutta, S. Bapat, V. Kulathumani, H. Zhang, V. Naik, V. Mittal, H. Cao, M. Demirbas, M. Gouda, Y. Choi, T Herman, S. Kulkarni, U. Arumugam, M. Nesterenko, A. Vora, and M. Miyashita. A line in the sand: a wireless sensor network for target detection, classification, and tracking. *Computer Networks*, 46(5):605–634, Dec 2004.

[8] D. Bailey, D. Coffin, A. Elbirt, J. Silverman, and A.Woodbury. NTRU in constrained devices. In Ç. Koç, D. Naccache, and C. Paar, editors, *Workshop on Cryptographic Hardware and Embedded Systems—CHES 2001*, volume 2162 of *Lecture Notes in Computer Science (LNCS)*, pages 266–277, Berlin, May 2001. Springer-Verlag.

[9] M. Bellare, R. Canetti, and H. Krawczyk. Keying hash functions for message authentication. In Neal Koblitz, editor, *Advances in Cryptology, Crypto '96*, volume 1109 of *Lecture Notes in Computer Science (LNCS)*, pages 1–15. Springer Verlag, 1996.

[10] Mihir Bellare, Joe Kilian, and Phillip Rogaway. The security of the cipher block chaining message authentication code. *Journal of Computer and System Sciences*, 61(3):362–399, 2000.

[11] Luca Benini and Giovanni De Micheli. System-level power optimization: Techniques and tools. *ACM Transactions on Design Automation of Electronic Systems*, 5(2):115–192, April 2000.

[12] Luca Benini, Giovanni De Micheli, and Enrico Macii. Designing low-power circuits: Practical recipes. *IEEE Circuits and Systems Magazine*, 1(1):6–25, 2001.

[13] F. Bennett, D. Clarke, J. B. Evans, A. Hopper, A. Jones, and D. Leask. Piconet: Embedded mobile networking. *IEEE Personal Communications*, 4(5):8–15, Oct 1997.

[14] John Black, Shai Halevi, Hugo Krawczyk, Ted Krovetz, and Phillip Rogaway. UMAC: Fast and secure message authentication. In *Advances in Cryptology - CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science (LNCS)*, pages 216–233. Springer-Verlag, 1999.

[15] M. Borah, R.M. Owens, and M.J. Irwin. Transistor sizing for low power CMOS circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 15(6):665–671, Jun 1996.

[16] S.M. Brennan, A.M. Mielke, D.C. Torney, and A.B. Maccabe. Radiation detection with distributed sensor networks. *Computer*, 37(8):57–59, Aug 2004.

[17] R. Burne et al. Self-organizing cooperative sensor network for remote surveillance: improved target tracking results. In *Proceedings of the SPIE - The International Society for Optical Engineering*, volume 4232, pages 313–321, Boston, 2001. SPIE, SPIE-Int. Soc. Opt. Eng, USA.

[18] C. Burwick, D. Coppersmith, E. D'Avignon, R. Gennaro, S. Halevi, C. Jutla, S. M. Matyas Jr., L. O'Connor, M. Peyravian, D. Safford, and N. Zunic. Mars - a candidate cipher for AES. In *First Advanced Encryption Standard (AES) Conference*, Ventura, California, USA, 1998.

[19] D. W. Carman, P. S. Kruus, and B. J. Matt. Constraints and approaches for distributed sensor network security. Technical report, NAI Labs, Security Research Division, Glenwood, MD, Sep 2000.

[20] L. Carter and M. Wegman. Universal classes of hash functions. *Journal of Computer and System Sciences*, 18(2):143–154, Apr 1979.

[21] Alberto Cerpa, Jeremy Elson, Deborah Estrin, Lewis Girod, Michael Hamilton, and Jerry Zhao. Habitat monitoring: application driver for wireless communications technology. *SIGCOMM Comput. Commun. Rev.*, 31(2 supp):20–41, 2001.

[22] Haowen Chan, A. Perrig, and D. Song. Random key predistribution schemes for sensor networks. In *Symposium on Security and Privacy, 2003*, pages 197–213, May 2003.

[23] A.P. Chandrakasan, S. Sheng, and R.W. Brodersen. Low-power CMOS digital design. *IEEE Journal of Solid-State Circuits*, 27(4):473–484, Apr 1992.

[24] Chee-Yee Chong and P. Kumar, Srikanta. Sensor networks: Evolution, opportunities, and challenges. *Proceedings of the IEEE*, 91(8):1247–1256, Aug 2003.

[25] Margery Conner. Energy harvesters extract power from light, vibrations. *EDN*, pages 45–50, Oct 27 2005. `http://www.edn.com/article/CA6275407.html`.

[26] Michael Crichton. *Prey*. HarperCollins, 2002.

[27] D. Culler, D. Estrin, and M. Srivastava. Guest editors' introduction: Overview of sensor networks. *Computer*, 37(8):41–79, Aug 2004.

[28] David E. Culler and Wei Hong. Wireless sensor networks. *Commun. ACM*, 47(6):30–33, Jun 2004.

[29] David E. Culler and Hans Mulder. Smart sensors to network the world. *Scientific American*, pages 84–91, Jun 2004.

[30] Vivek De and Shekhar Borkar. Technology and design challenges for low power and high performance [microprocessors]. In *International Symposium on Low Power Electronics and Design (ISLPED) 1999*, pages 163–168, 1999.

[31] S. Devadas and S. 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.

[32] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, Nov 1976.

[33] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, Jul 1985.

[34] Michael Epstein, Laszlo Hars, Raymond Krasinski, Martin Rosner, and Hao Zheng. Design and implementation of a true random number generator based on digital circuit artifacts. In Colin D. Walter, Çetin K. Koç, and Christof Paar, editors, *Proceedings of the 5th International Workshop on Cryptographic Hardware and Embedded Systems, CHES 2003*, volume 2779 of *Lecture Notes in Computer Science (LNCS)*, pages 152–165, Berlin, Sep 2003. Springer-Verlag.

[35] Laurent Eschenauer and Virgil D. Gligor. A key-management scheme for distributed sensor networks. In *Proceedings of the 9th ACM conference on Computer and communications security*, pages 41–47. ACM Press, 2002.

[36] 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 (LNCS)*, pages 234–251, New York, 1999. Springer-Verlag.

[37] M. Feldhofer, J. Wolkerstorfer, and V. Rijmen. AES implementation on a grain of sand. *Information Security, IEE Proceedings*, 152(1):13–20, Oct 2005.

[38] Martin Feldhofer, Sandra Dominikus, and Johannes Wolkerstorfer. Strong authentication for RFID systems using the AES algorithm. In Marc Joye and

Jean-Jacques Quisquater, editors, *Proceedings of the 6th international workshop on cryptographic hardware and embedded systems CHES 2004*, volume 3156 of *Lecture Notes in Computer Science (LNCS)*, pages 357–370. Springer, Aug 2004.

[39] Benjamin Fulford. Sensors gone wild. Forbes Global, Oct 2002. `http://www.forbes.com/global/2002/1028/076_print.html`.

[40] Prasanth Ganesan, Ramnath Venugopalan, Pushkin Peddabachagari, Alexander Dean, Frank Mueller, and Mihail Sichitiu. Analyzing and modeling encryption overhead for sensor network nodes. In *Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications*, pages 151–159, 2003.

[41] Gunnar Gaubatz, Jens-Peter Kaps, Erdinç Öztürk, and Berk Sunar. State of the art in ultra-low power public key cryptography for wireless sensor networks. In *Third IEEE International Conference on Pervasive Computing and Communications Workshops, Workshop on Pervasive Computing and Communications Security–PerSec'05*, pages 146–150. IEEE Computer Society, Mar 2005.

[42] Gunnar Gaubatz, Jens-Peter Kaps, and Berk Sunar. Public key cryptography in sensor networks—revisited. In Hannes Hartenstein, Claude Castellucia, Christof Paar, and Dirk Westhoff, editors, *1st European Workshop on Security in Ad-Hoc and Sensor Networks (ESAS 2004)*, volume 3313 of *Lecture Notes in Computer Science (LNCS)*, pages 2–18, Heidelberg, August 2004. Springer.

[43] A. Ghosh, S. Devadas, K. Keutzer, and J. White. Estimation of average switching activity in combinational and sequential circuits. In *Proceedings of the* 29^{th} *Design Automation Conference*, pages 253–259, Jun 1992.

[44] P. Girard, C. Landrault, S. Pravossoudovitch, and D. Severac. A gate resizing technique for high reduction in power consumption. In *ISLPED '97: Proceedings of the 1997 international symposium on Low power electronics and design*, pages 281–286, New York, NY, USA, 1997. ACM Press.

[45] Tim Good and Mohammed Benaissa. AES on FPGA from the fastest to the smallest. In Josyula R. Rao and Berk Sunar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2005*, volume 3659 of *Lecture Notes in Computer Science (LNCS)*, pages 427–440. Springer, 2005.

[46] James Goodman and Anantha P. Chandrakasan. Low power scalable encryption for wireless systems. *Wireless Networks*, 4(1):55–70, Jan 1998.

[47] P.F. Gorder. Sizing up smart dust. *Computing in Science & Engineering*, 5(6):6–9, Nov.-Dec. 2003.

[48] Tim Grembowski, Roar Lien, Kris Gaj, Nghi Nguyen, Peter Bellows, Jaroslav Flidr, Tom Lehman, and Brian Schott. Comparative analysis of the hardware implementations of hash functions SHA-1 and SHA-512. In Agnes Hui Chan and Virgil Gligor, editors, *Information Security, 5th International Conference, ISC 2002*, volume 2433 of *Lecture Notes in Computer Science (LNCS)*, pages 75–89. Springer-Verlag, 2002.

[49] 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 (LNCS)*, pages 172–189. Springer, 1997.

[50] H. Handschuh and D. Naccache. SHACAL. Submission to the NESSIE project, Gemplus, F-92447 Issy-les-Moulineaux, France, Oct 2000.

[51] H. Handschuh and D. Naccache. SHACAL: a family of block ciphers. Submission to the NESSIE project, Gemplus, F-92447 Issy-les-Moulineaux, France, 2001.

[52] Helena Handschuh, Lars R. Knudsen, and Matthew J. Robshaw. Analysis of SHA-1 in encryption mode. In David Naccache, editor, *Topics in Cryptology CT-RSA 2001*, volume 2020 of *Lecture Notes in Computer Science (LNCS)*, pages 70–83. Springer Verlag, 2001.

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

[54] Jason Hill, Mike Horton, Ralph Kling, and Lakshman Krishnamurthy. The platforms enabling wireless sensor networks. *Commun. ACM*, 47(6):41–46, Jun 2004.

[55] Jason Hill, Robert Szewczyk, Alec Woo, Seth Hollar, David E. Culler, and Kristofer S. J. Pister. System architecture directions for networked sensors. In *Architectural Support for Programming Languages and Operating Systems*, pages 93–104. ACM, 2000.

[56] Jason Lester Hill. *System Architecture for Wireless Sensor Networks*. Phd. dissertation, University of California at Berkeley, Spring 2003.

[57] J. Hoffstein and J. H. Silverman. Optimizations for NTRU. In *Proceedings of Public Key Cryptography and Computational Number Theory*. de Gruyter, Warsaw, September 2000.

[58] Jeffrey Hoffstein, Nick Howgrave-Graham, Jill Pipher, Joseph H. Silverman, and William Whyte. NTRUSign: Digital signatures using the NTRU lattice. In Marc Joye, editor, *Topics in Cryptology–CT-RSA 2003*, volume 2612 of *Lecture Notes in Computer Science*, pages 122–140, Heidelberg, April 2003. RSA, Springer Verlag.

[59] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. NTRU: A ring-based public key cryptosystem. In J.P. Buhler, editor, *Algorithmic Number Theory*

(ANTS III), volume 1423 of *Lecture Notes in Computer Science (LNCS)*, pages 267–288, Berlin, Jun 1998. Springer-Verlag.

[60] Jeffrey Hoffstein, Joseph H. Silverman, and William Whyte. NTRU report 012, version 2. estimated breaking times for NTRU lattices. Technical Report 12, NTRU Cryptosystems, Inc., Burlington, MA, USA, June 2003.

[61] Fei Hu and Neeraj K. Sharma. Security considerations in ad hoc sensor networks. *Ad Hoc Networks*, 3(1):69–89, Jan 2005.

[62] P. Ienne and M.A. Viredaz. Bit-serial multipliers and squarers. *IEEE Transactions on Computers*, 43(12):1445–1450, Dec 1994.

[63] International Technology Roadmap for Semiconductors. *ITRS Executive Summary*, 2005 edition. `http://public.itrs.net/`.

[64] Don Johnson, Alfred Menezes, and Scott Vanstone. The elliptic curve digital signature algorithm (ECDSA). *International Journal of Information Security*, 1(1):36–63, Aug 2001.

[65] J. M. Kahn, R. H. Katz, 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.

[66] B. Kaliski. The MD2 message-digest algorithm. RFC 1319, RSA Laboratories, Apr 1992.

[67] Jens-Peter Kaps and Berk Sunar. Energy comparison of AES and SHA-1 for ubiquitous computing. In Xiaobo Zhou et al., editor, *Embedded and Ubiquitous Computing (EUC-06) Workshop Proceedings*, Lecture Notes in Computer Science (LNCS). Springer, 2006. to appear.

[68] Jens-Peter Kaps, Kaan Yüksel, and Berk Sunar. Energy scalable universal hashing. *IEEE Transactions on Computers*, 54(12):1484–1495, Dec 2005.

[69] Chris Karlof, Naveen Sastry, and David Wagner. TinySec: A link layer security architecture for wireless sensor networks. In *Second ACM Conference on Embedded Networked Sensor Systems (SenSys 2004)*, pages 162–175, New York, 2004. ACM Press.

[70] Ralph Kling, Robert Adler, Jonathan Huang, Vincent Hummel, and Lama Nachman. Intel mote-based sensor networks. *Structural Control and Health Monitoring*, 12(3-4):469–479, 2005.

[71] N. Koblitz. Hyperelliptic cryptosystems. *Journal of Cryptology*, 1(3):139–150, 1989.

[72] Neal Koblitz. Elliptic curve cryptosystems. *Mathematics of Computation*, 48(177):203–209, Jan 1987.

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

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

[75] H. Krawczyk, M. Bellare, and R. Canetti. HMAC: Keyed-hashing for message authentication. RFC 2104, Network Working Group, Feb 1997.

[76] X. Lai and J. L. Massey. A proposal for a new block encryption standard. In Ivan B. Damgård, editor, *Advances in Cryptology - EuroCrypt '90*, volume 473

of *Lecture Notes in Computer Science (LNCS)*, pages 389–404, Berlin, 1990. Springer-Verlag.

[77] Jeremy Landt. The history of RFID. *IEEE Potentials*, 24(4):8–11, 2005.

[78] Y.W. Law, J. Doumen, and P. Hartel. Benchmarking block ciphers for wireless sensor networks. In *IEEE International Conference on Mobile Ad-hoc and Sensor Systems*, pages 447–456. IEEE, Oct 2004.

[79] Arjen K. Lenstra and Eric R. Verheul. Selecting cryptographic key sizes. *Journal of Cryptology: The Journal of the International Association for Cryptologic Research*, 14(4):255–293, 2001.

[80] Philip Levis and David Culler. Maté: 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.

[81] Xiaohua Luo, Kougen Zheng, Yunhe Pan, and Zhaohui Wu. Encryption algorithms comparisons for wireless networked sensors. In *IEEE International Conference on Systems, Man and Cybernetics*, volume 2, pages 1142–1146. IEEE, Oct 2004.

[82] Alan Mainwaring, Joseph Polastre, Robert Szewczyk, David Culler, and John Anderson. Wireless sensor networks for habitat monitoring. In *First ACM Workshop on Wireless Sensor Networks and Applications*, Atlanta, GA, USA., Sep 2002.

[83] Stefan Mangard, Manfred Aigner, and Sandra Dominikus. A highly regular and scalable AES hardware architecture. *IEEE Transactions on Computers*, 52(4):483–491, April 2003. Special Issue on Cryptographic Hardware and Embedded Systems.

[84] 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.

[85] M. Maroti, G. Simon, A. Ledeczi, and J. Sztipanovits. Shooter localization in urban terrain. *Computer*, 37(8):60–61, Aug 2004.

[86] K. Martinez, J.K. Hart, and R. Ong. Environmental sensor networks. *Computer*, 37(8):50–56, Aug 2004.

[87] R. J. McEliece. *Finite Fields for Computer Scientists and Engineers.* Kluwer Academic Publishers, 2nd edition, 1989.

[88] MDx-MAC and Building Fast MACs from Hash Functions. Preneel, b. and van oorschot, p. c. In Don Coppersmith, editor, *Advances in Cryptology, Crypto '95*, volume 963 of *Lecture Notes in Computer Science (LNCS)*, pages 1–14. Springer-Verlag, 1995.

[89] S. Meininger, J.O. Mur-Miranda, R. Amirtharajah, A.P. Chandrakasan, and J.H. Lang. Vibration-to-electric energy conversion. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 9(1):64–76, Feb 2001.

[90] A. J. Menezes, P. C. van Oorschot, and S. Vanstone. *Handbook of Applied Cryptography.* CRC Press Inc., 1997.

[91] V. S. Miller. Uses of elliptic curves in cryptography. In Hugh C. Williams, editor, *Advances in Cryptology — CRYPTO '85*, volume 218 of *Lecture Notes in Computer Science (LNCS)*, pages 417–426, Berlin, 1986. Springer-Verlag.

[92] National Institute of Standards and Technology (NIST), FIPS Publication 81. *DES modes of operation*, Dec 1980. `http://csrc.nist.gov/publications/fips/fips81/fips81.htm`.

[93] National Institute of Standards and Technology (NIST), FIPS Publication 113. *Computer Data Authentication*, May 1985. http://www.itl.nist.gov/fipspubs/fip113.htm.

[94] National Institute of Standards and Technology (NIST), FIPS Publication 185. *Escrowed Encryption Standard (EES)*, Feb 1994. http://www.itl.nist.gov/fipspubs/fip185.htm.

[95] National Institute of Standards and Technology (NIST). *Complete SKIPJACK and KEA specification*, Jun 1998. http://csrc.nist.gov/CryptoToolkit/skipjack/skipjack-kea.htm.

[96] National Institute of Standards and Technology (NIST), FIPS Publication 46-3. *Data Encryption Standard (DES)*, Oct 1999. http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf.

[97] National Institute of Standards and Technology (NIST), FIPS Publication 197. *Advanced Encryption Standard (AES)*, Nov 2001. http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf.

[98] National Institute of Standards and Technology (NIST), FIPS Publication 198. *The Keyed-Hash Message Authentication Code (HMAC)*, Mar 2002. http://csrc.nist.gov/publications/fips/fips198/fips-198a.pdf.

[99] 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.

[100] Wim Nevelsteen and Bart Preneel. Software performance of universal hash functions. In *EUROCRYPT'99*, volume 1592 of *Lecture Notes in Computer Science (LNCS)*, pages 24–41, Berlin, 1999. Springer-Verlag.

[101] E. Öztürk, B. Sunar, and E. Savaş. Low-power elliptic curve cryptography using scaled modular arithmetic. In Marc Joye and Jean-Jacques Quisquater, editors, *Workshop on Cryptographic Hardware and Embedded Systems–CHES 2004*, volume 3156 of *Lecture Notes in Computer Science (LNCS)*, pages 92–106. Springer, Aug 2004.

[102] Erdinç Öztürk. Low power elliptic curve cryptography. Msc in electrical and computer engineering, Worcester Polytechnic Institute, Worcester, Massachusetts, USA, Apr 2005.

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

[104] Massoud Pedram. Power minimization in IC design: Principles and applications. *Transactions on Design Automation on Electronic Systems TODAES*, 1(1):3–56, Jan 1996. Tutorial and Survey Paper.

[105] Massoud Pedram and Jan Rabaey. *Power Aware Design Methodologies.* Kluwer Academic Publishers, Norwell, Massachusetts, 2002.

[106] Jan Pelzl, Thomas Wollinger, Jorge Guajardo, and Christof Paar. Hyperelliptic curve cryptosystems: Closing the performance gap to elliptic curves. In Colin D. Walter, Çetin K. Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems–CHES 2003*, volume 2779 of *Lecture Notes in Computer Science (LNCS)*, pages 351–365, Berlin, September 2003. Springer Verlag.

[107] A. Perrig, R. Canetti, J.D. Tygar, and Dawn Song. Efficient authentication and signing of multicast streams over lossy channels. In *IEEE Symposium on Security and Privacy, 2000*, pages 56–73, May 2000.

[108] 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.

[109] Adrian Perrig, John Stankovic, and David Wagner. Security in wireless sensor networks. *Commun. ACM*, 47(6):53–57, Jun 2004.

[110] Joseph Polastre. Design and implementation of wireless sensor networks for habitat monitoring. Master's thesis, University of California at Berkeley, Spring 2003.

[111] Joseph Polastre, Robert Szewczyk, and David Culler. Telos: Enabling ultra-low power wireless research. In *The Fourth International Conference on Information Processing in Sensor Networks: Special track on Platform Tools and Design Methods for Network Embedded Sensors (IPSN/SPOTS)*, pages 364–369, April 2005.

[112] P. Prasithsangaree and Prashant Krishnamurthy. Analysis of energy consumption of RC4 and AES algorithms in wireless LANs. In *IEEE Global Telecommunications Conference, GLOBECOM '03*, volume 3, pages 1445–1449. IEEE, Dec 2003.

[113] Bart Preneel. The state of cryptographic hash functions. In I. Damgård, editor, *Lectures on Data Security: Modern Cryptology in Theory and Practice*, volume 1561 of *Lecture Notes in Computer Science (LNCS)*, pages 158–182. Springer-Verlag, 1999.

[114] J.M. Rabaey and M. Pedram. *Low Power Design Methodologies.* Kluwer Academic Publishers, Norwell, Massachusetts, 1996.

[115] M. O. Rabin. Digitalized signatures and public key functions as intractable as factorization. MIT/LCS/TR-212, Massachusetts Institute of Technology, Jan 1979.

[116] 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.

[117] K. Ren, K. Zeng, and W. Lou. On broadcast authentication in wireless sensor networks. In *International Conference on Wireless Algorithms, Systems, and Applications (WASA 2006)*, Xi'an, China, Aug 2006. to appear.

[118] R. Rivest. The MD4 message-digest algorithm. RFC 1320, MIT Laboratory for Computer Science and RSA Data Security Inc., Apr 1992.

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

[120] R. Rivest, M. J. B. Robshaw, R. Sidney, and Y. L. Yin. The RC6$^{\text{TM}}$ block cipher. In *First Advanced Encryption Standard (AES) Conference*, Ventura, California, USA, 1998.

[121] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, Feb 1978.

[122] R.L. Rivest. The RC5 encryption algorithm. In B. Preneel, editor, *Fast Software Encryption*, volume 1008 of *Lecture Notes in Computer Science (LNCS)*, pages 86–96, Berlin, 1995. Springer-Verlag.

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

[124] K. Roy, S. Mukhopadhyay, and H. Mahmoodi-Meimand. Leakage current mechanisms and leakage reduction techniques in deep-submicrometer CMOS circuits. *Proceedings of the IEEE*, 91(2):305–327, 2003.

[125] Markku-Juhani O. Saarinen. Cryptanalysis of block ciphers based on SHA-1 and MD5. In Thomas Johansson, editor, *Fast Software Encryption, 10th International Workshop, FSE 2003*, volume 2887 of *Lecture Notes in Computer Science (LNCS)*, pages 36–44, Feb 2003.

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

[127] Sanjay E. Sarma, Stephen A. Weis, and Daniel W. Engels. RFID systems and security and privacy implications. In Burton S. Kaliski Jr., Çetin K. Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2002*, volume 2523 of *Lecture Notes in Computer Science (LNCS)*, pages 454–469, Aug 2002.

[128] B. Schneier, J. Kelsey, D. Whiting, D. Wagner, C. Hall, and N. Ferguson. Twofish: A 128-bit block cipher. AES proposal, Counterpane Systems, Minneapolis, MN, USA, June 1998.

[129] Bruce Schneier. *Applied Cryptography.* John Wiley & Sons, 2nd edition, 1995.

[130] 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 (LNCS)*, pages 74–85, New York, 1996. Springer-Verlag.

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

[132] Frank Stajano and Ross Anderson. The resurrecting duckling: Security issues for ad-hoc wireless networks. In B. Christianson, B. Crispo, and M. Roe, editors, *Security Protocols, 7th International Workshop*, Berlin, Heidelberg, 1999. Springer Verlag.

[133] D. R. Stinson. Universal hashing and authentication codes. In Joan Feigenbaum, editor, *Advances in Cryptology - CRYPTO '91*, volume 576 of *Lecture Notes in Computer Science (LNCS)*, pages 74–85, 1992.

[134] D. R. Stinson. *Cryptography: Theory and Practice.* CRC Press, Boca Raton, 1995.

[135] Douglas R Stinson. *Cryptography: Theory and Practice*, volume 36 of *Discrete Mathematics and its Applications.* Chapman & Hall/CRC, Boca Raton, 3rd edition, 2005.

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

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

[138] Synopsys Inc. *Power Compiler User Guide*, release 2004.06 edition, Jun 2004.

[139] Robert Szewczyk, Eric Osterweil, Joseph Polastre, Michael Hamilton, Alan Mainwaring, and Deborah Estrin. Habitat monitoring with sensor networks. *Commun. ACM*, 47(6):34–40, Jun 2004.

[140] J. J. Thomas, J. M. Keller, and G. N. Larsen. The calculation of multiplicative inverses over GF(p) efficiently where p is a mersenne prime. *IEEE Transactions on Computers*, 5(35):478–482, 1986.

[141] S. Turgis, N. Azemard, and D Auvergne. Explicit evaluation of short circuit power dissipation for CMOS logic structures. In *Proceedings of the 1995 International Symposium on Low Power Design*, pages 129–134, 1995.

[142] H.J.M Veendrick. Short-circuit dissipation of static CMOS circuitry and its impact on the design of buffer circuits. *IEEE Journal of Solid-State Circuits*, 19(4):468–473, Aug 1984.

[143] Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. Collision search attacks on SHA1. Internet, Feb 2005.

[144] R Want. Enabling ubiquitous sensing with RFID. *Computer*, 37(4):84–86, 2004.

[145] B. Warneke, M. Last, B. Liebowitz, and K.S.J. Pister. Smart dust: communicating with a cubic-millimeter computer. *Computer*, 34(1):44–51, Jan 2001.

[146] Brett A. Warneke. *Ultra-Low Energy Architectures and Circuits for Cubic Millimeter Distributed Wireless Sensor Networks*. Phd. dissertation, University of California at Berkeley, Spring 2003.

[147] Brett A. Warneke and Kristofer S.J. Pister. An ultra-low energy microcontroller for smart dust wireless sensor networks. In *IEEE International Solid-State Circuits Conference, 2004*, 2004. Digest of Technical Papers.

[148] M. Wegman and L. Carter. New hash functions and their use in authentication and set equality. *Journal of Computer and System Sciences*, 22(3):265–279, Jun 1981.

[149] A. Weimerskirch, C. Paar, and S. Chang Shantz. Elliptic curve cryptography on a palm os device. In Y. Mu V. Varadharajan, editor, *The 6th Australasian Conference on Information Security and Privacy (ACISP 2001)*, volume 2119 of *Lecture Notes in Computer Science (LNCS)*, pages 502–513, Heidelberg, Jul 2001. Springer-Verlag.

[150] Stephen A. Weis. Security and privacy in radio-frequency identification devices. Master's thesis, Massachusetts Institute of Technology, May 2003. Master's Thesis.

[151] Stephen A. Weis, Sanjay E. Sarma, Ronald L. Rivest, and Daniel W. Engels. Security and privacy aspects of low-cost radio frequency identification systems. In *1st Annual Conference on Security in Pervasive Computing*, Mar 2003. `http://www.dfki.de/SPC2003/`.

[152] David Wheeler and Roger Needham. TEA extensions. Technical report, Cambridge University, England, Oct 1997.

[153] Kaan Yüksel, Jens-Peter Kaps, and Berk Sunar. Universal hash functions for emerging ultra-low-power networks. In *Proceeding of The Communications Networks and Distributed Systems Modeling and Simulation Conference (CNDS)*, San Diego, CA, January 2004.

[154] Sencun Zhu, Shouhuai Xu, S. Setia, and S. Jajodia. Establishing pairwise keys for secure communication in ad hoc networks: a probabilistic approach. In *11th IEEE International Conference on Network Protocols*, pages 326–335, Nov 2003.

Appendix A

References for Cryptographic Algorithms

A.1 Block Ciphers

- DES/3DES [96]
- IDEA [76]
- RC5 [122]
- AES (Rijndael) [97]
- RC6 [120]
- MARS [18]
- Serpent [6]
- Twofish [128]

A.2 Stream Cipher

- RC4, description can be found in [129]

A.3 Hash Functions

- MD2 [66]
- MD4 [118]
- MD5 [119]
- Secure Hash Standard (SHS) [99]
- NH [14]
- WH [153]
- see also [113] and [133]

A.4 Public Key Cryptosystems

- RSA [121]
- ElGamal [33]
- Rabin's Scheme [115]
- Elliptic curve cryptography (ECC) [72, 91]
- Hyperelliptic curve cryptography (HECC) [71]
- NtruEncrypt system [59]