2009

Data Integrity Protection For Security in Industrial Networks

Mohsen Bahramali

Data Integrity Protection For Security in Industrial Networks

(Thesis Format: Integrated Article)

by

Mohsen Bahramali

Faculty of Engineering Science
Department of Electrical and Computer Engineering

Submitted in partial fulfillment
of the requirements for the degree of
Master of Engineering Science

School of Graduate and Postdoctoral Studies
The University of Western Ontario
London, Ontario, Canada
December, 2009

Abstract

Modern industrial systems are increasingly based on computer networks. Network-based control systems connect the devices at the field level of industrial environments together and to the devices at the upper levels for monitoring, configuration and management purposes. Contrary to traditional industrial networks which are considered stand-alone and proprietary networks, modern industrial networks are highly connected systems which use open protocols and standards at different levels. This new structure of industrial systems has made them vulnerable to security attacks. Among various security needs of computer networks, data integrity protection is the major issue in industrial networks. Any unauthorized modification of information during transmission could result in significant damages in industrial environments.

In this thesis, the security needs of industrial environments are considered first. The need for security in industrial systems, challenges of security in these systems and security status of protocols used in industrial networks are presented. Furthermore, the hardware implementation of the Secure Hash Algorithm (SHA) which is used in security protocols for data integrity protection is the main focus of this thesis. A scheme has been proposed for the implementation of the SHA-1 and SHA-512 hash functions on FPGAs with fault detection capability. The proposed scheme is based on time redundancy and pipelining and is capable of detecting permanent as well as transient faults. The implementation results of the proposed scheme on Xilinx FPGAs show small area and timing overhead compared to the original implementation without fault detection. Moreover, the implementation of SHA-1 and SHA-512 on Wireless Sensor Boards has been presented taking into account their memory usage and execution time. There is an improvement in the execution time of the proposed implementation compared to the previous works.

Keywords: Industrial Networks, Data Integrity Protection, Hash Function, Secure Hash Algorithm, Fault Detection, Wireless Sensor Boards.

Statement of Co-Authorship

I herby declare that this thesis incorporates two original papers one of which were published in a conference proceeding and the second one is to be submitted for publication in a peer reviewed journal. These papers were co-authored by my supervisors, Dr.Jin Jiang and Dr. Arash Reyhani Masoleh and the collaboration is covered in Chapters 2 and 3 of the thesis. In all cases, the key ideas, primary contributions, experimental designs, data analysis and interpretation, were of the author, and the contribution of the co-author was primarily through the provision of the research problem, reviews, supervision, and guidance throughout the project.

I am aware of the University of Western Ontario policies on authorship and I certify that I have properly acknowledged the contribution of all co-authors to my thesis. I certify that, with the above qualification, this thesis, and the research to which it refers, is the product of my own work.
The papers included in this thesis are as follow:

- Chapter 2: **M. Bahramali**, J. Jiang, A. Reyhani Masoleh, "Security Issues in Industrial Control Systems," *NPIC-HMIT 2009 - Nuclear Plant Instrumentation, Control, and Human-Machine Interface Technologies,* April 5-9, 2009

- Chapter 3: **M. Bahramali**, *J. Jiang, A. Reyhani Masoleh,* "A Fault Detection Scheme For SHA-1 and SHA-512 Hash Functions," *Submitted to: IEEE Transactions on Very Large Scale Integration (VLSI) Systems,* 2010.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Modern industrial systems are based on computer networks. Network-based control systems are being used in various industries. From the very top level where management, configuration and monitoring data exist to the field level where sensors, actuators and controllers such as PLCs (Programmable Logic Controllers) are located, the date is transferred through the networks. While general IT networks are used in the top levels of industrial systems, the bottom levels utilize special kinds of networks designed based on the needs of industrial environments. These networks usually transfer real time data among the devices and require stringent timing and high reliability. The network-based structure of modern industrial systems has made them vulnerable to security attacks as opposed to the traditional industrial systems which were considered stand-alone networks and security was not an issue in them. This thesis is concerned with the security needs of industrial networks.

1.1 Security in Communication Systems

Talking about network security, several issues should be taken into account. The need for security in a network stems from the fact that it can be vulnerable to security attacks. An attack is any action that compromises the security of a network and takes advantage of the information transferred through the network either by accessing the confidential data or modifying the critical data etc. On the other hand, a security

service is any action that decreases the vulnerability of a network against security attacks. The security objectives vary depending on the network and the application in which it is being used. The security objectives can be categorized as follows:

Confidentiality: This objective deals with the protection of the data transmitted in a network from an attacker. Using confidentiality services, an unauthorized person would not have access to the data transferred through the network. Other than the content of a message transmitted, the source, destination, frequency, length or the time of transmission might be required to be kept secret. Encryption and decryption mechanisms are usually used for confidentiality purposes.

Integrity: Integrity services deal with the prevention of data from modification during transmission. The purpose of this service is to make sure that the data is transmitted correctly and without unauthorized modification. It should also prevent delay in the transmission of the message, message injection or message replication. This service is very important in industrial systems where crucial information is transferred through the network. In some cases, violating the integrity may cause safety issues.

Authentication and Authorization: The purpose of authentication services is to make sure that the message received by a recipient is from the source it claims to be from. This service is concerned with the true identity of the communication parties. By using authentication services, authorization mechanisms limit the access to the network. Using this service, no unauthorized person can have access to the host system of a network.

Nonrepudiation: This service is concerned with the accountability and liability of the system. It prevents the sender from denying a transmitted message. It also prevents the recipient from denying a received message. Using this service, the recipient can prove that the message was sent by the alleged source and vice versa. Digital signatures are examples of this service.

Availability: The aim of this service is to make sure that the system is accessible by an authorized person all the time. There are several kinds of attacks which target the availability of the system. They aim to cause the system to stop working. The

availability is very important in industrial networks which are used for crucial applications. Any stop in the operation of these systems can result in a lot of damages.

The services mentioned above are the main objectives of the security mechanisms used in communication networks. Depending on the application, some of these services may be used to provide the system with the security required. For example, in the field level of industrial networks, the confidentiality of the data transmitted is not a big issue while the data integrity is very important. Any unauthorized modification in the information transferred among sensors, actuators and controllers may result in severe damage to the system.

An attacker violates one of these security services to take advantage of the system. Depending on the type of services being violated, there are several kinds of attacks. For example, Denial-of-Service (DoS) attacks violate the availability of the system. This can be done by flooding fake messages to the network. Eavesdropping attacks violate the confidentiality of the system while a man-in-the-middle attack violates both confidentiality and data integrity. Other kinds of attacks such as viruses and trojan horses can violate authentication, authorization, confidentiality and also availability of the system.

Cryptographic algorithms are used to provide a system with the security services mentioned above against the security attacks. Although they can not satisfy all kinds of security objectives, they are the main countermeasures against security attacks. They can provide confidentiality, data integrity, authentication and non-repudiation. For authorization and availability issues, some other issues other than cryptography should be taken into account. Cryptographic algorithms fall into three main categories:

Secret Key Cryptographic algorithms use a single key for encryption and decryption of the messages. The key should be transferred securely between sender and receiver. Secret key algorithms are fast algorithms used to encrypt and decrypt large amount of information. Block ciphers and stream ciphers are two kinds of secret key algorithms. Block ciphers encrypt and decrypt the messages in blocks of data while stream ciphers process the messages byte-by-byte or bit-by-bit. Advanced Encryption

Standard (AES) and Data Encryption Standard (DES) are examples of block ciphers and RC4 is a byte-wise stream cipher.

The second type of cryptographic algorithms are **public key** algorithms. Public key algorithms use two different keys to encrypt and decrypt messages. A public key is used to encrypt (decrypt) the message and a secret key is used to decrypt (encrypt) the transferred message. Using public key algorithms, there is no need to transfer a key between communication parties thus removing the key distribution problem existing in secret key cryptography. By their nature, public key algorithms are mathematically complex algorithms that require a large amount of processing. This makes their implementation very slow and resource consuming. They are used to transmit the "secret key" in secret key cryptography and also in digital signature schemes. RSA and ECC (Elliptic Curve Cryptography) are examples of public key algorithms.

Message Authentication Codes (MACs) and **One-way Hash Functions** are the other types of cryptographic protocols which are used for integrity protection purposes. They usually convert an arbitrary size message into a fixed-size data block and are used as cryptographic checksums to provide the integrity of the message. Message authentication codes use a key for this purpose while there is no key involved in hash functions. The MD5 and SHA (Secure Hash Algorithm) are examples of hash functions used extensively in security protocols.

Secret key and public key algorithms are used to encrypt/decrypt messages for confidentiality purposes and digital signature schemes. On the other hand, hash functions and message authentication codes are used for data integrity purposes. As mentioned previously, confidentiality of messages transferred in an industrial environment are not very important compared to their integrity. Data integrity is one of the main security issues in industrial environments. Therefore, hash functions and message authentication codes are the main cryptographic protocols that must be considered in these environments.

1.2 Data Integrity and Hash Functions

Hash functions are used for data integrity purposes in communication systems. Data integrity or message authentication services prevent unauthorized modification in the content of the messages, timing of messages, sequence of messages etc. There are three general methods to provide a system with message authentication or data integrity: Encryption, Message Authentication Codes and Hash Functions. In encryption, the ciphertext of a message serves as the authenticator; the sender encrypts the message with a shared key and the receiver uses the same key to decrypt the message. If the recipient can retrieve the original message from the received one, he/she can make sure that the message has not been altered during transmission. This is only true when the transmitted message follows a certain pattern. Otherwise, the recipient can not differentiate between the original message and the altered one.

Message Authentication Code is another method of providing data integrity. The sender calculates the MAC of the message using a secret key and appends it to the message. The message and its MAC are sent through the network. The recipient receives the message, calculates the MAC of the message using the same secret key. If the calculated MAC and the received MAC are identical, the recipient can be sure that the message has been sent correctly and without modification. If an attacker modifies the message during transmission, the calculated MAC at the receiving side would not be the same as the received MAC and the receiver would find out that the message has been modified during transmission. This is because of the fact that a secret key is used to calculate the message authentication code and the attacker does not have access to the key.

Another cryptography algorithm which is used for integrity purposes is hash function. Hash functions are public functions that map an arbitrary-length input to a fixed-length output which is used as the authenticator. Hash functions are usually used with encryption/decryption algorithms to provide message authentication services. The way a hash function can be used along with encryption/decryption algorithms to provide a message authentication scheme is shown in Figure 1.1. Figure 1.1

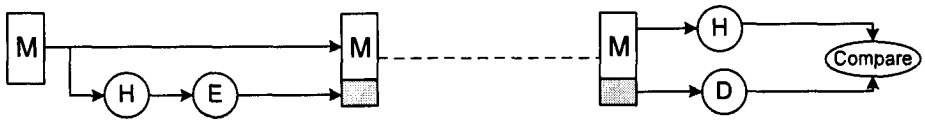


Figure 1.1: Message Authentication using Hash Functions

shows that the hash value of the message is encrypted and appended to the message by the sender. The receiver calculates the hash value of the received message. He/she also decrypts the received hash value and compares it to the calculated hash value. If these two hash values are identical, the receiver makes sure that the message has not been altered during transmission. In this method, because only the hash value of the message is encrypted and decrypted, it reduces the processing burden required to encrypt the whole message.

A cryptographic hash function should have certain properties:

- It should be easy to calculate the output.
- Given the output, it should be computationally difficult to calculate the input.
- A single bit change in the input should result in a significant change (more than half of the bits) in the output of the function.

Hash functions are the building blocks of many cryptographic algorithms and are used in a lot of security protocols.

Current most widely used hash functions are different versions of the the Secure Hash Algorithm (SHA) recommended by NIST [2]. The SHA-1 [2] hash function has a 160-bit output, whereas SHA-2 [2] has three different outputs of 256, 384 and 512 bits denoted by SHA-256, SHA-384 and SHA-512 respectively. The SHAs are currently used in several security protocols such as IPSec [3] and Secure Socket Layer (SSL) [1].

1.3 Hash Function Implementation

Like other cryptographic functions, the hardware implementation of hash functions is of great importance. Because of their speed, FPGAs are suitable platforms for the implementation of cryptographic algorithms. To improve the performance of the SHA hash functions in terms of speed, area utilization and power consumption, several techniques have been used in the literature [4], [5], [6], [7], [8], [9], [10], [11]. These techniques mainly include using Carry-Save-Adders (CSAs), unrolling, pipelining and operation rescheduling.

Addition is the main operation in the SHA structure. Any improvement in the design of adders can result in a significant improvement in the SHA design. Carry-Save-Adders increase the speed of the addition operation when three or more operands are to be added. They have been used in many structures to improve the speed of SHA-1 and SHA-512 functions [4], [5], [6]. The unrolling method is used to unroll multiple rounds of the SHA function into one round [7], [8]. The speed improvement in the unrolling method stems from the fact that the delay of the unrolled K rounds of the SHA structure would be less than K times the delay of the single-round structure. Moreover, the pipeline method can be used to make the SHA implementations running at higher clock frequencies [9], [10], [12], [13]. Because of the iterative structure of SHA hash functions, the pipeline method can efficiently increase the speed of the circuit. The combination of pipelining and loop-unrolling has also been used in the SHA implementation [12]. Operation rescheduling is another method of increasing the speed of SHA implementations [11]. In these implementations, part of the round computation is performed in the previous round leading to a decrease in the critical path of the round.

1.4 Thesis Outline

In this thesis, hardware implementation of the Secure Hash Algorithm (SHA) which is the most widely used hash function in security protocols will be considered for

industrial applications. The thesis contains three main chapters:

1.4.1 Security in Industrial Networks

Chapter 2 is a of about the security in industrial networks. It discusses the main reasons that have made security issues important in modern industrial networks as opposed to the traditional industrial networks in which security was not an issue. Modern industrial networks are moving towards a Complete CIM (Computer Integrated manufacturing) model in which devices of different layers are connected together. This large interconnection of devices and also using IP-based and wireless protocols in industrial networks has made them vulnerable to security attacks which was not the case for old industrial systems. This chapter also discusses the differences between industrial networks and the general IT networks from the security point of view and mentions the challenges of providing industrial networks with security mechanisms. The issues such as the real-time requirements of control networks, high reliability and resource limitations of embedded devices used in industrial environments are reviewed. Because security was not an issue when the protocols for industrial networks were designed, they do not have strong security mechanisms. This chapter also discusses the security status of the protocols currently used in industrial networks and shows their weaknesses compared to the security needs of modern industrial networks.

1.4.2 A Fault Detection Scheme For the FPGA implementation of SHA-1 and SHA-512 Hash Functions

Because data integrity is crucial in industrial networks, hash functions are very important in these environments. **Chapter 3** is concerned with the FPGA implementation of the Secure Hash Algorithm (SHA). The Secure Hash Algorithm recommended by NIST is the most widely used hash function in current security protocols. The SHA-1 hash function has a 160-bit output, whereas SHA-2 has three different outputs of 256, 384 and 512 bits. The SHAs are currently used in several security protocols such as IPSec and Secure Socket Layer (SSL). Like other cryptographic functions,

the hardware implementation of hash functions is of great importance. To improve the performance of the SHA hash functions in terms of speed, area utilization and power consumption, several techniques have been used in the literature. In this chapter, a method is introduced for the implementation of SHA-1 and SHA-512 hash functions which has the fault detection capability. Because of the iterative structure of hash functions, a single error in their hardware implementation could result in a large number of errors in the final hash value. In this chapter, we propose a fault detection scheme for the FPGA implementation of SHA-1 and SHA-512 round computation. The proposed fault detection scheme is based on time redundancy for involutional functions. It can detect permanent as well as transient faults as opposed to the normal time redundancy technique which is only capable of detecting transient errors. We use the pipelining method along with time redundancy to overcome the timing overhead created by the time redundancy technique. The proposed design does not impose significant timing overhead on the original implementation of SHA-1 and SHA-512 round computation. We have implemented the proposed scheme on Xilinx FPGAs to evaluate our design in terms of area and timing overhead.

1.4.3 Implementation of SHA Hash Functions on Wireless Sensor Board

Chapter 4 deals with the implementation of SHA hash functions on wireless sensor boards. Wireless sensor networks are extensively used in different applications. Industrial systems use wireless sensor networks for monitoring and configuration purposes. Because of the wireless communication in these kinds of networks, they are vulnerable to security attacks. On the other hand, due to the hardware constraints existing in the sensor nodes, it is usually very challenging to apply security mechanisms to wireless sensor networks. Sensor nodes usually suffer from limited computational power, limited memory and limited power supply. Cryptographic algorithms which are used in security mechanisms are usually complex algorithms which require a significant amount of memory and processing power. This makes it quite challenging

to implement cryptography algorithms on wireless sensor boards. In this chapter, we implement the SHA-1 and SHA-2 hash functions on sensor boards and compare their implementations in terms of memory requirements and execution time. This comparison will be helpful in choosing these hash functions for a specific application.

1.5 Contributions

The contributions of this thesis can be briefly described as follows:

- A new fault detection scheme for the FPGA implementation of SHA-1 and SHA-512 is introduced in this thesis. The main feature of this scheme is that although it is based on time redundancy, it is capable of detecting permanent faults as well as transient faults.
- To the best of our knowledge, this is the first FPGA implementation of the SHA-1 hash function which has the fault detection capability. Because of using the pipelining method, the proposed design has a very small (3%) timing overhead.
- There is an improvement in the timing overhead of the proposed scheme for SHA-512 compared to the previous works. The timing overhead of the proposed design is 10 percent while two previous works show 11.6 percent and 73 percent timing overhead.
- This fault detection scheme is capable of detecting any kind of faults as opposed to the parity-based schemes which only detect faults with an odd number of erroneous bits.
- The sensor board implementation of SHA-1 done in this thesis shows execution time and ROM usage improvement compared to the previous work.
- To the best of our knowledge, this is the only implementation of SHA-512 on sensor boards.

Bibliography

[1] W. Stallings, *Cryptography and network security*, Prentice Hall Upper Saddle River, NJ, 2003.

[2] National Institute of Standards and Technology, "Secure Hash Standard (SHS), FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION, FIPS PUB 180-3", (2008).

[3] RFC 2401, "The Security Architecture for the Internet Protocol", (1998).

[4] L. Dadda, M. Macchetti and J. Owen, "The design of a high speed ASIC unit for the hash function SHA-256 (384, 512)," in *Proc. Design, Automation and Test in Europe Conference and Exhibition*, 2004.

[5] T. Grembowski, R. Lien, K. Gaj, N. Nguyen, P. Bellows, J. Flidr, T. Lehman and B. Schott, "Comparative Analysis of the Hardware Implementations of Hash Functions SHA-1 and SHA-512," in *Proc. 5th International Conference on Information Security, ISC*, 2002, pp. 75-89.

[6] M. Macchetti and L. Dadda, "Quasi-pipelined hash circuits," in *Proc. IEEE Symposium on Computer Arithmetic*, 2005, pp. 222-229.

[7] F. Crowe, A. Daly, T. Kerins and W. Marnane, "Single-chip FPGA implementation of a cryptographic co-processor," in *Proc. IEEE International Conference on Field-Programmable Technology*, pp. 279-285, 2004.

[8] R. Lien, T. Grembowski, and K. Gaj, "A 1 Gbit/s Partially Unrolled Architecture of Hash Functions SHA-1 and SHA-512," *Springer J. CT-RSA*, pp. 324-338, 2004.

[9] N. Sklavos, G. Dimitroulakos and O. Koufopavlou, "An ultra high speed architecture for VLSI implementation of hash functions," in *Proc. 10th IEEE International Conference on Electronics, Circuits and Systems, ICECS,* 2003, vol. 3, pp. 990-993.

[10] A.P. Kakarountas, G. Theodoridis, T. Laopoulos and C.E. Goutis, "High-Speed FPGA Implementation of the SHA-1 Hash Function," *in Proc. IEEE Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS,* 2005, pp. 211-215.

[11] R. Chaves, G. Kuzmanov, L. Sousa and S. Vassiliadis, "Cost-Efficient SHA Hardware Accelerators," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems,* vol.16, no.8, pp.999-1008, 2008.

[12] H. Michail, A.P. Kakarountas, O. Koufopavlou and C.E. Goutis, C.E., "A low-power and high-throughput implementation of the SHA-1 hash function," in *Proc. IEEE International Symposium on Circuits and Systems, ISCAS* 2005. Vol. 4, pp. 4086-4089.

[13] G. Wang, "An Efficient Implementation of SHA-1 Hash Function," in *Proc. IEEE International Conference on Electro/information Technology,* 2006, pp.575-579.

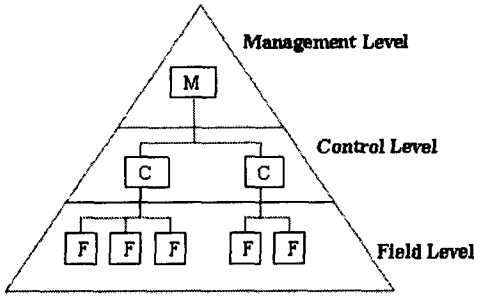


Figure 2.1: CIM(Computer Integrated Manufacturing Architecture)

toring and configuration workstations and carries the control data among them. The top level which is called the management level is used for higher level management of different sections of the network and can be viewed as a plantwide local area network linking office workplaces. Figure 2.2 shows a realization of this hierarchy. Depending on the application, control and field level may be merged together and directly connected to the management level. The management level could be connected to the Internet for web-based applications. While general IT protocols such as LAN and WAN protocols are used in the management level of control networks, the control level and field level of these networks have special protocols. Contrary to traditional control networks which usually used vendor specific protocols and mechanisms, modern control networks have been moving towards standardization of protocols used at different levels of the network. Standard protocols are widely used at the field level and control level of current control networks.

Traditional control networks were considered isolated networks. They were usually small networks in which there was not a large amount of interconnectivity in the devices inside the network and also there was no connection to the outside world. This is not the case for modern control networks as depicted in Figure 2.2. Although this architecture facilitates the easy communication of devices at different levels, remote configuration and monitoring of devices, easier maintenance etc, it has created some other issues, one of the most important of which is security. As a matter of fact, whenever there is a great amount of interconnection in a system, security issues come

out. There are some specific features such as the stringent timing requirements which make it difficult to apply general security mechanisms used in general IT networks to industrial networks. These requirements should be taken into account when designing a security mechanism for industrial environments. This paper aims to look into the security issues and challenges in industrial networks. The rest of this paper is organized as follows: In Section 2, the need for security in modern control networks is studied. Section 3 discusses the challenges in providing the industrial networks with security mechanisms. The status of current protocols used in industrial networks in terms of security are studied in Section 4 and the paper concludes in Section 5.

2.2 The Need for Security in Industrial Control Networks

There are many reasons which verify the need for security measures in modern control networks: Firstly, as mentioned before, the amount of interconnection available in such networks has made them vulnerable to security attacks. Contrary to traditional control networks, many devices from different levels of the network and also from outside networks can get connected to the systems inside the network. Without proper security strategies such as authentication and access control, they can make modifications in the devices which might result in an incident and cause a great damage to the entire network. Security vulnerabilities in industrial environments could also cause safety issues.

On the other hand, standard protocols used in different layers of modern industrial networks, even in the field level, have made them more vulnerable in terms of security. Control industry used to use proprietary devices and protocols which made the control devices vendor specific. As a result, only few people had the technical knowledge of the system and the communication mechanism among the devices. This made those systems more immune to security incidents because except for a few people, nobody knew about the technical details of the system and could not take advantage of any

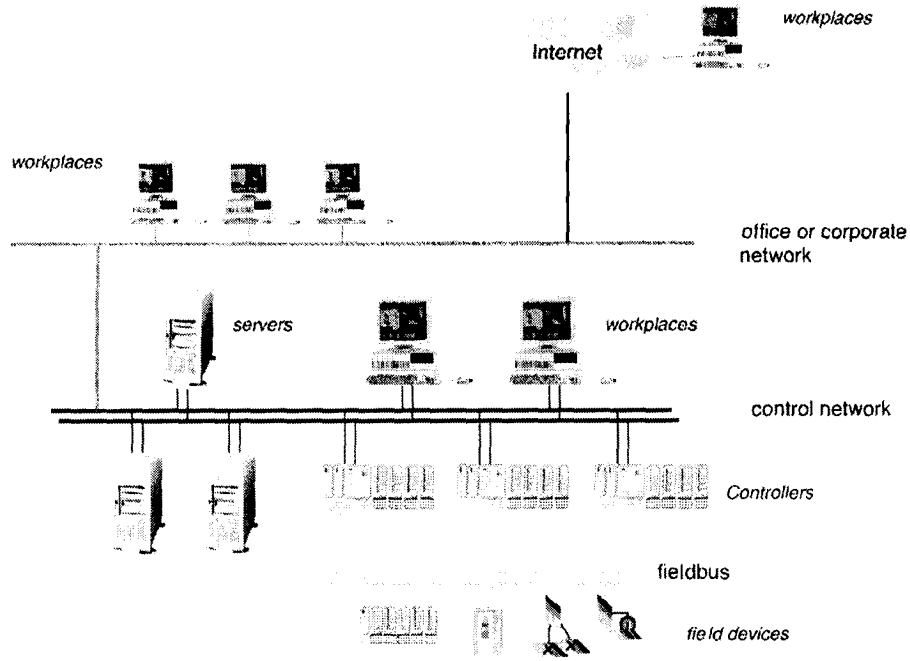


Figure 2.2: CIM(Architecture of Modern Control Networks)

vulnerability in the system. It is always said that traditional control networks had the "security by obscurity" policy. They were secure because they were obscure. This policy can no longer be effective in control networks. To benefit from interoperability, easy maintenanace etc. current industrial networks are using standard protocols and mechanisms which are public and everybody can be aware of them. That's the reason why "security by obscurity" policy can not be used in modern control networks.

Furthermore, because of the ubiquity of the Internet Protocol and its widespread use in different applications, control networks are moving towards using IP based protocols. The top level of control networks in the CIM pyramid is a general IT network which is usually a kind of TCP/IP network such as LAN or WAN. TCP/IP-based protocols are used in the control level of current control networks as well. These protocols use TCP/IP as their transport layer and their application layer sits above the TCP or UDP. Foundation Fieldbus HSE is a good example of this kind of protocols. TCP/IP based protocols are even getting into the field level of control networks. ProfiNet is an example of such protocols which is an IP-based version of the

PROFIBUS [3] network. Using TCP/IP-based protocols in control networks makes them more vulnerable to security attacks because:

1. TCP/IP protocol stack is renowned for its security vulnerabilities and there are lots of known attacks on IP networks.

2. TCP/IP is a widely used network and there are lots of people and organizations who are familiar with its vulnerabilities and therefore, successful attacks are much more likely in these kinds of networks.

These facts make the control networks vulnerable to the general IP attacks and cause security issues to gain importance in them.

Another important issue is using wireless protocols in industrial environments. With the success of wireless technology in data and voice communication systems, control networks are moving towards the wireless technology as well. Although it is not a long time that wired control networks have gained a widespread attention and they are the first choice in industrial environments, it seems that they will be superseded by wireless control networks in the near future. The success of wireless communication in other areas such as data and media transmission accelerates this process. The idea of connecting sensors, actuators and controllers through a wireless link motivates the automation companies to consider wireless technology very seriously. There are inherent benefits in a wireless industrial system. Removing cable from the plants makes the installation of devices much easier and decreases the setup time. It also simplifies the maintenance of the plant compared to the wired technologies. Easier access to the field level for diagnostic and maintenance purposes will be possible in wireless technology. Handheld devices can be remotely connected to the field devices temporarily; this reduces the fault detection and correction time. Whether or not current wireless technologies can be applied to fieldbus systems is the main concern of wireless control networks. Fieldbus networks have some timing and reliability requirements which cannot be provided using the current wireless technologies. To do this, some modifications and special investigations are required.

Although fieldbus networks are moving towards the wireless fieldbuses, because

of the widespread use of wired fieldbus technology in various automation industries it cannot be superseded by wireless technology in a short time. Some middle stages are required to perform this transition completely. So a combination of current wired fieldbus systems and wireless technology is usually used in the field level of industrial networks [4]. There has been some research on wireless fieldbuses in recent years [5], [4] and a few have been implemented [5]. By their nature, wireless systems are more vulnerable to the threats and attacks and strict security measures should be taken in wireless environments. It is evident that physical access to the network in a wireless system is much easier than a wired network and therefore, a security attack to a wireless network would be simpler.

The issues mentioned above are the main reasons why security has become so important in current industrial networks. The security incidents reported in recent years justify this claim. In January 2003, the Slammer worm penetrated the network of Ohio's Davis-Besse nuclear power plant and affected its safety monitoring system and caused it to be shut down for about 5 hours [6]. The communication system of a USA transportation company was infected by a virus in August 2003 and caused the freight and passenger transportation system to be halted and made difficulties in the morning commuter traffic [7]. An Australian man was sent to prison accused of hacking into the Maroochy Shire, Queensland's waste management system in March 2000. He caused millions of litres of sewage to spill out in the surrounding areas [8]. These incidents show that the security threats to control networks are real and they can cause severe problems. There are some other reported incidents such as the Washington gasoline pipeline failure in 1999 and the Northeast power blackout in 2003. The British Columbia Institute of Technology (BCIT) maintains a database called Industrial Security Incident Database (ISID) which keeps record of security incidents in industrial environments. According to this database, the number of recorded security incidents between 1982 and 2006 is depicted in Figure 2.3 [9]. This diagram indicates a rapid increase in the number of security incidents from 2001 which reaches its highest point in 2003 and 2004. This implies the importance of security in current industrial networks. The decrease in the number of incidents in

2005 and 2006 is probably because of the awareness of the industry of the importance of security and taking some countermeasures in these years.

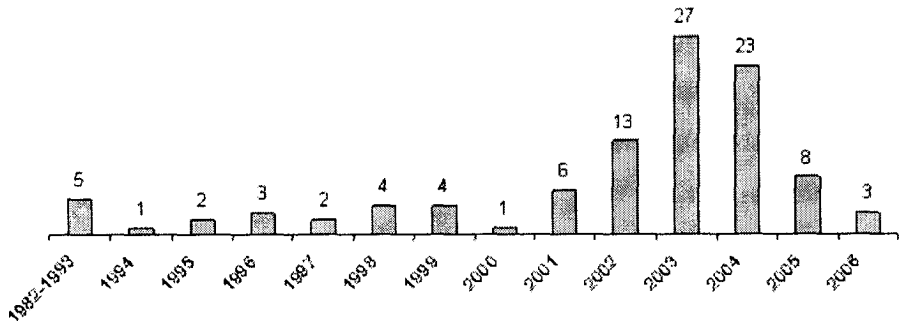


Figure 2.3: Number of Security Incidents by Year [9]

2.3 Challenges of Security in Industrial Networks

Unlike industrial control networks, security has always been an issue in general IT systems and several mechanisms and strategies have been used in various information networks. Considering security in control networks, one might think of applying the general security strategies used in IT systems to the industrial networks. Unfortunately this is not a viable solution. There are major differences between IT networks and industrial control systems which make it so difficult to use security strategies of IT networks in industrial control systems. These differences stem from the nature of control networks and industrial environments. The amount of reliability, safety and availability, timing constraints etc have made control networks much different from other kinds of networks such as the Internet. Some of the differences between control and IT networks are as follows:

Industrial control systems need a high degree of availability. Because control networks are usually used in critical industries such as process control industries, they need to be highly reliable. An unscheduled shutdown of the system is not accepted in such networks. Responses such as rebooting the system which is common

in IT networks are not acceptable in industrial networks. This level of availability necessitates a very strict testing of any security measure which is supposed to be deployed in the system.

Performance requirement is another difference between these two kinds of systems. Industrial systems usually require real time communication in which the packets should be sent and received at exact time intervals and time delay or packet loss is not accepted in these systems. These timing requirements and real time communications are not so stringent in many other kinds of networks. Because of their nature, cryptographic algorithms used in various security applications usually need a large amount of processing time and cause some delay in the communication of the data in the network which may not be acceptable in control networks.

High reliability is another feature of control networks which makes them distinct from IT networks. Industrial networks are usually used in critical environments such as petrochemical plants, power plants etc. and any failure in these systems could result in huge financial losses or even damage to the people and environment. This is not the case in general networks. Therefore, any security measure designed for industrial control networks should not decrease the reliability of the system. For example a password-based authentication system should not get in the way of the normal operation of the system in the case of an emergency.

Another difference which should be considered in designing security strategies for control networks is the component lifetime. Due to the rapid evolution of technology, typical IT networks have a lifetime of 3 to 5 years. For control networks, because of the types of application they are being used for and the costs of changing components, the lifetime goes up to 15 to 20 years which makes it more difficult to adopt an appropriate security mechanism for this lifetime.

Unlike IT systems, components of control networks such as sensors, actuators and controllers are usually implemented in embedded systems which do not have lots of processing power and resources such as memory and power supply. Because of the complexity of cryptographic algorithms in terms of processing power and memory requirements, this resource constraint is a very important issue in applying security

mechanisms to control networks.

As the final point, it should be mentioned that security objectives in control networks and IT networks could be different. For example, confidentiality of data transmitted is the major point in many IT networks while this can not usually be an issue in control networks. In contrast, integrity of data transmitted and resistance to Denial of Service attacks (DoS) is of greatest importance in control networks.

The issues mentioned above imply that there are some major differences between IT networks and control networks which make it impossible to directly apply the security strategies of IT networks to control networks. The differences in the requirements of these two kinds of networks and also the limitations available in industrial networks should be seriously taken into consideration for adopting a security mechanism for such networks.

2.4 Security Status of the Protocols Used in Industrial Control Networks

Security in industrial networks can be viewed from two standpoints:

- Security in control and management level
- Security in the field level

There are several mechanisms and standards in the control and management level of industrial networks. Some of these mechanisms and standards are: OPC [10], MMS [11], IEC 61850 [12] and ICCP [13]. The security status of these standards are described as follows [1]:

1. OPC Security: OPC is a standard interface for data communication among components of an automation network. It is a high level standard by which the communication of devices will be independent of their lower layers' architecture. Field devices in an automation networks can use OPC to transfer data to the operation workstations. They contain an optional access control mechanisms based on Access

Control Lists (ACLs). There is not any confidentiality or integrity support in the OPC Standarad.

2. MMS is an application layer standard for communication between field devices and PLCs. It is mainly used in manufacturing automation systems. It is usually implemented over TCP/IP and it has a password based access control mechanism with no confidentiality or integrity.

3. IEC 61850: This Standard is used in electric power networks. It does not have any network security mechanism and relies on the security measures of lower layers such as IPSec.

4. ICCP: ICCP is another application layer data communication standard which is used in power plants. Because it is usually implemented over TCP/IP, it uses SSL sessions for security purposes. It also uses an SSL-Based mechanism in non-TCP/IP implementations. The security vulnerabilities of ICCP have been discussed in [14].

On the other hand, whether to apply security mechanisms in the field level is an issue in industrial networks. The current status of the protocols used in the field level of control networks shows some weak security mechanisms in them. In [15], current state of the main fieldbus systems used in modern control networks in terms of security has been studied. According to this study, Foundation fieldbus has password and access group mechanisms in its application layer (FMS in H1 protocol and FDA in HSE protocol stack). Although there exists an authentication mechanism, it seems to be more for QoS services rather than security purposes. Similarly, The Profibus system (Profibus PA, Profibus DP and Profinet) has a basic access protection mechanism. ControlNet has a password based authentication in its data link layer which is used only locally and is not transmitted over the network. Its application layer also has an authentication mechanism for identifying devices. World-FIP and Interbus have access rights and also 8 bit plain-text passwords and P-Net shows a simple write protection mechanism. These mechanisms are weak compared to the requirements of current systems because:

- They do not use any encryption or protection mechanisms for the passwords

- Length of passwords is very short and easy to break

- They only provide authentication mechanisms and confidentiality which may be necessary in some applications has been ignored completely

- The mechanisms are not mandatory to be implemented in the standard.

It is evident that security has not been taken into consideration seriously in the design of these protocols and they need some add-on services and procedures to be considered as secure systems. This lack of security is mainly because of the fact that security was not an important issue when these protocols were designed. This is not true for all fieldbus protocols. The protocols used in Building Automation Systems (BAS) such as LonWorks, EIB and BACNet do usually have more efficient security strategies. Especially BACNet contains a DES-based security system which seems to be the only fieldbus protocol which has used a serious security mechanism.

Because of the increasing importance of security for control networks, there have been some field level security implementations in recent years. [16] has developed a secure fieldbus protocol. This protocol is based on DES to encrypt and decrypt the fieldbus messages. It uses automatic key exchange and key refresh mechanisms to provide the field devices with the appropriate keys. [17] has proposed using IPSec for IP based fieldbus networks. This approach has some advantages. First, using IPSec which operates at the IP layer, there would be no need to make any modification in the fieldbus protocol. Besides, IPSec has already been used in other applications and this could reduce the amount of time and effort required to make the fieldbus network secure. The BACNet standard mentioned before is another example of using security in the field level. It contains the confidentiality and data integrity services which are based on the DES cryptographic algorithm and a trusted key server. [18] has extended this approach and offered an AES-based cryptography mechanism for building automation systems.

2.5 Conclusion

Network-based control systems are widely used in industrial environments. The increasing interconnection of devices at the different levels of control networks make them vulnerable to security attacks. Using standard protocols in modern industrial networks as opposed to the proprietary protocols used in old networks, using IP-based protocols at different layers in the CIM hierarchy and also wireless communication in the field level of these networks are the main reasons that security has become important in industrial control networks.

Industrial networks usually require stringent real-time communication and also a high degree of reliability which make it difficult to apply IT security mechanisms to them. Therefore, some modifications are needed to provide the industrial networks with general IT security mechanisms. On the other hand, field devices used in industry are usually implemented in embedded systems which are limited in terms of memory and processing power. These limitations make it challenging to implement security mechanisms in field devices.

The Current status of protocols used in various layers of industrial networks show that security issues were not taken into account when these protocols were designed. There is a need to modify these protocols to meet the security requirements of modern industrial networks.

Bibliography

[1] D. Dzung, M. Naedele, T. P. Von Hoff, M. Crevatin, "Security for Industrial Communication Systems",*Proceedings of the IEEE* , vol.93, no.6, pp.1152-1177, June 2005.

[2] Fieldbus Foundation. "ff-581-1.3, foundation specification: System architecture", 2003.

[3] PROFIBUS International. "iec 61158, digital data communication for measurement and control - Fieldbus for use in industrial control systems", 1999.

[4] Dong-Hyuk Choi; Jung Il Lee; Dong-Sung Kim; Woo Chool Park, "Design and Implementation of Wireless Fieldbus for Networked Control Systems," *SICE-ICASE International Joint Conference* , 2006. vol., no., pp.1036-1040, 18-21 Oct. 2006

[5] Willig, A., Matheus, K.; Wolisz, A., "Wireless Technology in Industrial Networks," *Proceedings of the IEEE* , vol.93, no.6, pp.1130-1151, June 2005

[6] K Poulsen. Slammer worm crashed ohio nuke plant net, August 2003.

[7] Computer virus strikes csx transportation computers-freight and commuter service affected, August 2003.

[8] T Smith. Hacker jailed for revenge sewage attacks, October 2001

[9] Keith Stouffer, Joe Falco, and Karen Scarfone. Guide to industrial control systems (ics) security. Technical report, National Institute of Standards and Technology, September 2007.

[10] F. Iwanitz and J. Lange, OLE for Process Control. Heidelberg, Germany: Hthig, 2001.

[11] Industrial Automation SystemsManufacturing Message Specification(MMS), ISO 9506-1:2003, 9506-2:2003, 9506-5:1999, 9506-6:1994, 2003

[12] Communication Networks and Systems in Substations, Int. Std. IEC61850, 2003.

[13] Inter Control Centre Protocol (ICCP), Std. IEC 60 870-6 TASE.2,1997.

[14] ICCP (TASE.2) security enhancements, Electric Power Res. Inst.,Tech. Rep., Sep. 2003.

[15] Treytl A, Sauter T, and Schwaiger C. "Security measures for industrial fieldbus systems - state of the art and solutions for ip-based approaches". *IEEE International Workshop on Factory Communication Systems*, pages 201-209. IEEE, 2004.

[16] P Swaminathan, K Padmanabhan, S Ananthi, and R Pradeep. "The secure fieldbus (secfb) protocol - network communication security for secure industrial process control". *IEEE Region 10 Conference*, TENCON, pages 1-4. IEEE, November 2006.

[17] A Treytl, T Sauter, and C Schwaiger. "Security measures in automation systems-a practice-oriented approach", *IEEE Conference on Emerging Technologies and Factory Automation*, volume 2, pages 847-855, Sept 2005.

[18] W Granzer, W Kastner, G Neugschwandtner, and F Praus. "Security in networked building automation systems", *IEEE International Workshop on Factory Communication Systems*, pages 283-292, June 2006.

Chapter 3

A Fault Detection Scheme For SHA-1 and SHA-512 Hash Functions

3.1 Introduction

Message authentication is one of the most important security requirements of network security protocols. While Confidentiality services prevent the disclosure of data to an attacker, message authentication mechanisms deal with the modification (content, sequence, time, etc.) of the messages transmitted in a network. Message authentication also serves as a source repudiation countermeasure to verify that received messages come from the alleged source [1]. There are three types of mechanisms which can provide the authentication service [1]: Encryption, Message Authentication Codes (MACs) and Hash Functions. Using encryption, the encrypted message can be used as the authenticator. Message Authentication Codes are functions which produce a fixed-length output from the message and a secret key. This output can serve as the authenticator. Hash functions are public functions that map an arbitrary-length input to a fixed-length output which is used as the authenticator. Hash functions are usually used with encryption to provide message authentication services. The way

of using a hash function along with an encryption algorithm for authentication purposes has been described in [1]. A cryptographic hash function should have certain properties:

- It should be easy to calculate the output.
- Given the output, it should be difficult to calculate the input.
- A single bit change in the input should result in a significant change (more than half of the bits) in the output of the function.

Current most widely used hash functions are different versions of the Secure Hash Algorithm (SHA) recommended by NIST [2]. The SHA-1 [2] hash function has a 160-bit output, whereas SHA-2 [2] has three different outputs of 256, 384 and 512 bits denoted by SHA-256, SHA-384 and SHA-512 respectively. The SHAs are currently used in several security protocols such as IPSec [3] and Secure Socket Layer (SSL) [1]. Like other cryptographic functions, the hardware implementation of hash functions is of great importance. To improve the performance of the SHA hash functions in terms of speed, area utilization and power consumption, several techniques have been used in the literature [4], [5], [6], [7], [8], [9], [10] and [11]. These techniques mainly include using Carry-Save-Adders (CSAs), unrolling, pipelining and operation rescheduling.

Addition is the main operation in the SHA structure. Any improvement in the design of adders can result in a significant improvement in the SHA design. Carry-Save-Adders have been used in [4], [5] and [6] to increase the speed of SHA-1 and SHA-2 functions. The unrolling method is used in [7] and [8] to unroll multiple rounds of the SHA function into one round. The speed improvement in the unrolling method stems from the fact that the delay of the unrolled K rounds of the SHA structure would be less than K times the delay of the single-round structure. Moreover, the pipeline method has been used in [9], [10], [12] and [13] to make the structures running at higher clock frequencies. Because of the iterative structure of SHA hash functions, the pipeline method can efficiently increase the speed of the circuit. The combination of pipelining and loop-unrolling has been used in [12]. The high-speed

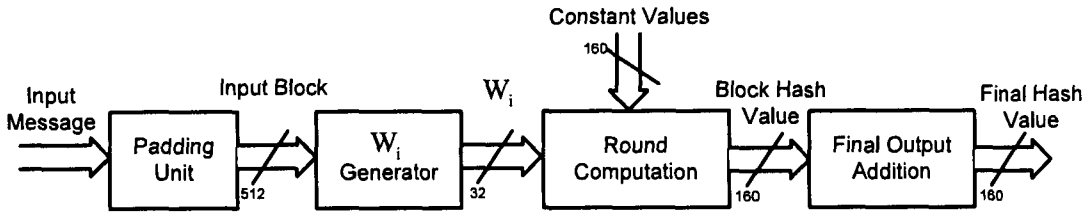
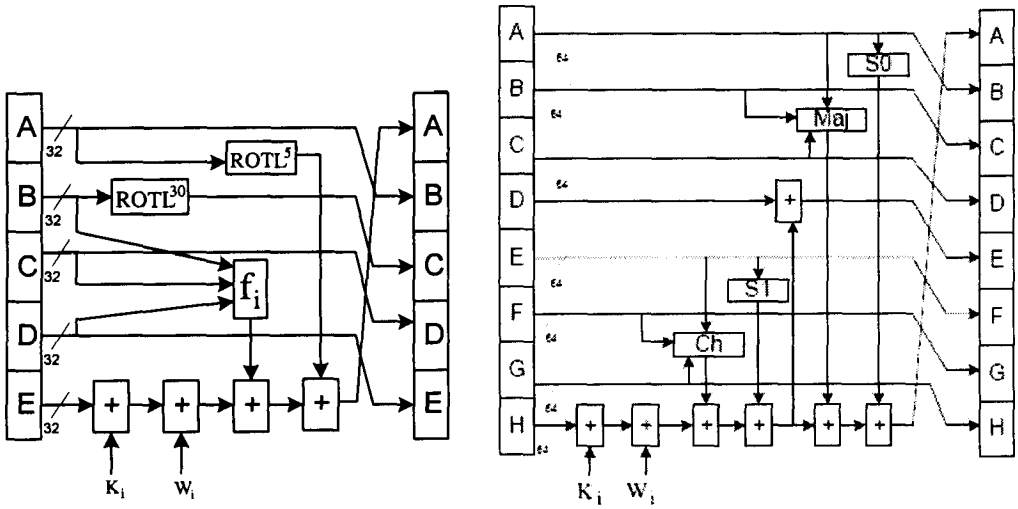


Figure 3.1: SHA-1 Architecture

implementations of the SHA-1 and SHA-2 which are based on operation rescheduling have been introduced in [11]. In these implementations, part of the round computation is performed in the previous round leading to a decrease in the critical path of the round.

Complexity of cryptographic algorithms makes their hardware implementations vulnerable to faults. There are various sources of faults such as temperature, white light, X-rays etc. which can cause a malfunction in a hardware implementation [14]. Concurrent Error Detection (CED) schemes have been extensively used in the implementation of cryptographic algorithms such as Advanced Encryption Standarad (AES) [17], [18] and [19]. Because of the inherent characteristics of hash functions mentioned above, a single fault in the round operation of hash functions will result in multiple error bits in the output hash value. Therefore, designing a reliable hash function in hardware is very important. In [15], an error analysis of the SHA-512 hash function has been done by injecting a fault at different stages of the hash value execution and investigating the propagation of such errors to the output. This has been done by introducing an error to the input of a single operation, input of a single round and the input of a block round. The frequency of output bit errors has been analyzed for each of the above cases. A parity based CED scheme for the SHA-512 hash function is also proposed and implemented in [15]. Several schemes have been introduced in [16] to apply error detection and correction to SHA-2 hash family. These techniques are mainly based on Hamming codes.

In general, there are three methods to provide a digital design with fault detection schemes: hardware redundancy, time redundancy and information redundancy. In the hardware redundancy techniques, two separate pieces of hardware are used to



(a) SHA-1 (b) SHA-512

Figure 3.2: SHA-1 and SHA-512 round computations

perform the same function and the outputs are compared to detect a fault in the design. This technique can detect both permanent and transient faults and provides 100 percent fault coverage. But it has a 100 percent resource overhead which is not suitable for many applications. In time redundancy techniques, a single hardware is reused at a different time to perform the same function. This technique is suitable for detecting transient faults and it has a 100 percent timing overhead. Information redundancy techniques use some additional information such as parity bits and other error detection codes to detect some kinds of faults in a system.

In this paper, we present a fault detection scheme for the implementation of the SHA-1 and SHA-512 hash functions. For this purpose, we use the time redundancy technique with different inputs to detect the faults in our design. Contrary to the normal time redundancy technique which performs the same function at different times, the proposed scheme performs the reverse of the SHA round computation for redundancy purposes. Therefore, this technique will be able to detect any transient or permanent faults in the SHA round computation design. It will also be able to detect arbitrary multiple-bit faults as opposed to [15] which is only capable of detection of an odd number of errors. Implementing the proposed design on FPGA shows that there

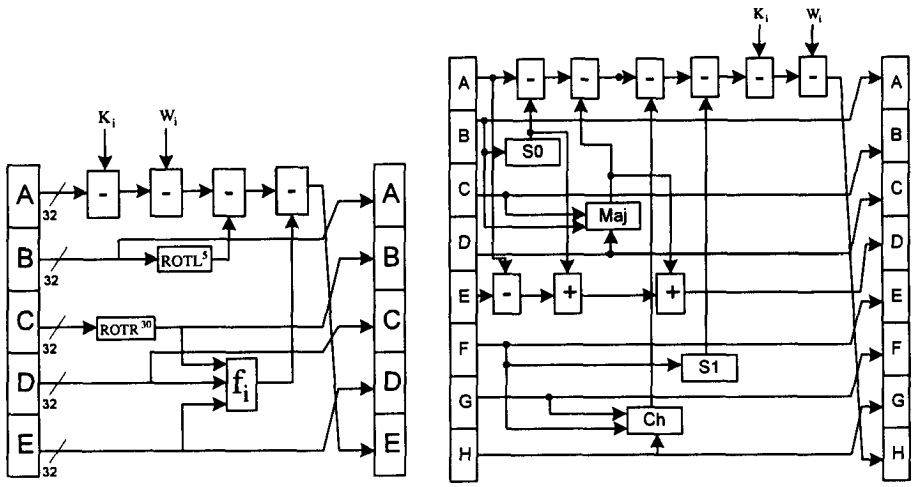
in Figure 3.1. The input message is split and padded into blocks of 512 bits. The block expansion unit generates 80 × 32-bit W_is from each block, one 32-bit for each round. The block diagram of the round computation has been shown in Figure 3.2(a). The round computation of the SHA-1 comprises of the addition, logical rotation and the function f_i. The round computation will be performed 80 times. As shown in Figure 3.2(a), A,B,C,D and E are 32-bit operands with constant initial values. The output of the last round will be added to the output of other blocks to generate the final hash value. All the additions are performed in modulo 2^{32}. There are four different functions f_i and four constants K_i, each for every 20 rounds. Theses functions and constants have been defined in Appendix A. The 80 W_is for each block of the input message are obtained from:

$$W_i = \begin{cases} M_i & 0 \leq i < 16, \\ RotL^1(W_{i-3} \bigoplus W_{i-8} \bigoplus W_{i-14} \bigoplus W_{i-16}) & 16 \leq i < 80. \end{cases} \tag{3.1}$$

where M_i denotes the ith 32-bit word of the input data block, $RotL^1$ is the one-bit circular shift to the left and $\oplus$ is the bit-wise XOR operation.

3.2.2 SHA-512

The SHA-512 is one of the three versions of the SHA-2 hash functions which generates a 512-bit hash value. The general structure of SHA-512 is the same as SHA-1 which is shown in Figure 3.1. The data sizes and the internal structure of units are different from SHA-1. The data blocks are 1024 (16×64) bits long and the the block expansion generates 64-bit W_is which are fed into the round computation. The round computation unit has been shown in Figure 3.2(b). Like the SHA-1 hash function, the round function for SHA-512 mainly consists of addition operations which are performed in modulo 2^{64} instead of modulo 2^{32} in SHA-1. Similarly, the A,B,C,...H and the K_i constants are 64-bit words. The round computation is performed 80 times in



(a) SHA-1 (b) SHA-512

Figure 3.4: SHA-1 and SHA-512 inverse round computation

SHA-512 and the 64-bit W_is are obtained from:

$$W_i = \begin{cases} M_i & 0 \leq i < 16, \\ \sigma_1(W_{i-2}) + W_{i-7} + \sigma_0(W_{i-15}) + W_{i-16} & 16 \leq i < 80 \end{cases} \tag{3.2}$$

The $+$ denotes the arithmetic addition operation performed in modulo 2^{64} and σ_0 and σ_1 functions are defined in Appendix A.

3.3 A Fault Detection Scheme For SHA-1 and SHA-512 Round Computations

In this section, we present fault detection schemes for the SHA-1 and SHA-512 hash functions. We propose a time redundancy technique for the round computation of SHA-1 and SHA-512 to detect their transient and permanent faults. For this purpose, their round computation is broken into two parts and a pipeline stage is inserted between these two parts. Each part is computed twice and the results are compared together for fault detection. Because of the pipeline stage added to the round, the

Table 3.1: Sequence of Operations for the proposed design

Clock Cycle (i)	Registers Operations	Round Operation1	Round Operation2
$i = 1$	$IR(i) \Leftarrow Input$	Add	-
$i = 2, 4, 6, \cdots$	$R(i) \Leftarrow IR(i-1)$ $CR(i) \Leftarrow R(i-1)$	Subtract	Add
$i = 3, 5, 7, \cdots$	$IR(i) \Leftarrow Output(i-1)$	Add	Subtract

maximum frequency will be doubled thus making the throughput almost the same as the original round without fault detection capability.

As seen from Figure 3.2, the most important operation in both SHA-1 and SHA-512 is the addition operation. We propose a time redundancy technique for the addition operations in these hash functions. Figure 3.3(a) shows a normal time redundancy technique applied to a single adder. $C(t_1)$ in this figure is the output of adding the operands A and B at time t_1. Similarly, A and B are added at time t_2 to result in $C(t_2)$. $C(t_1)$ and $C(t_2)$ are compared to detect any transient faults in the adder. Obviously, this method is not able to detect permanent faults because any permanent fault would result in the same faulty $C(t_1)$ and $C(t_2)$ thus would be undetected by this method. To overcome this problem, we use the method shown in Figure 3.3(b). As shown in this figure, A and B are added at time t_1. C and B are subtracted at time t_2 to result in $A(t_2)$. $A(t_1)$ and $A(t_2)$ are then compared to detect the faults in the adder. Because different inputs are used at time t_1 and t_2, this method would be able to detect permanent as well as transient faults. This method requires using an adder/subtractor instead of an adder which will be shown that it would not impose significant overhead in the design of SHA round computations.

To apply this time redundancy technique to the round computation of SHA-1 and SHA-512, we use the their inverse round computation. As shown in Figure 3.4, the inverse round computation uses subtraction instead of addition. The same circuit with minor modification can be used to perform both addition and subtraction, thus this method does not impose a significant overhead to the original circuit. The modification mainly includes replacing the adders with adders/subtractors.

Figure 3.5 shows the way the proposed time redundancy technique can be used in the SHA-1 structure. The round function for both SHA-1 and SHA-512 is performed

80 times to process a single block. The critical path of the round operation has been broken into two halves and a pipeline register has been added to the round function. There are four 160-bit registers in Figure 3.5. The input register (IR) and the pipeline register (PR) hold the input and intermediate values of the round computation while the R and CED register are used for comparison. The way these registers are loaded in each clock cycle to perform the round operation and fault detection is depicted in Table 3.1. First, the message block is fed into the input register. In the first two clock cycles ($i = 1, 2$), the input message goes through the first round of the hash function. The second round starts with the third clock cycle. In the fourth clock cycle, while the second half of the round computation is processing the second round, the first half starts the reverse computation. The first round of inverse computation ends in the fifth clock cycle which contains the first comparison. The third round of the hash function computation starts at the same clock cycle. The R and CED registers are used to store the input value of each round to be compared with the output of the inverse round computation. *Table 3.1 indicates that:*

- *The input register is loaded in odd clock cycles. The previous value of this register is used in even clock cycles.*
- The R and CED registers are loaded in even clock cycles to store the input value for comparison.
- The pipeline register is loaded in all clock cycles.

From Table 3.1, one can also find that the half round operations in Figure 3.5 should switch between addition and subtraction every other clock cycle. The first comparison will be done in the fifth clock cycle where the input to the first round is compared to the output of the first inverse round operation. This procedure will run 163 clock cycles to perform all 80 rounds of the hash function along with the redundancy check. The same architecture can be applied to the SHA-512 structure. The registers would be 512-bit registers instead of 160-bit registers for SHA-1. The next two sections will present the detailed structure of adding a pipeline stage to the round operation of SHA-1 and SHA-512 respectively.

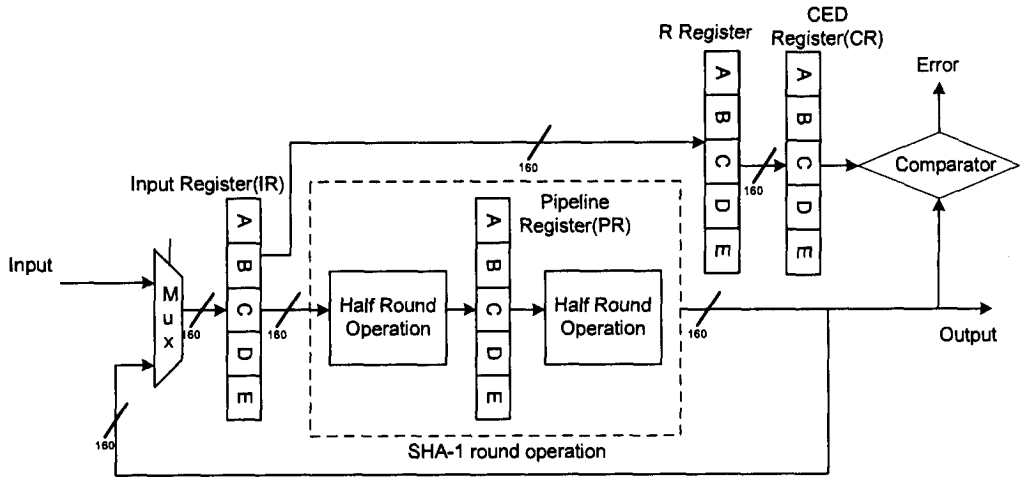


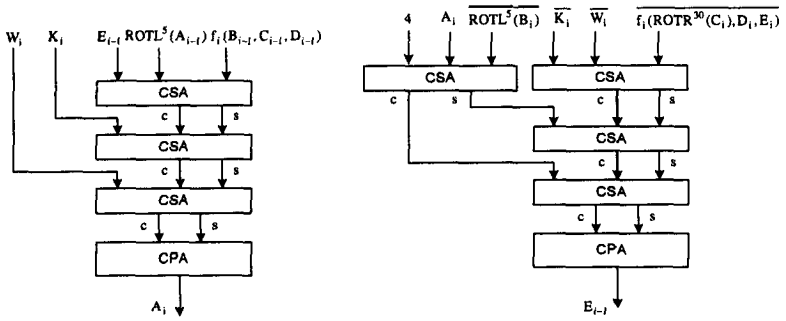
Figure 3.5: Time Redundancy for SHA-1 Hash Functions

3.3.1 SHA-1

The SHA-1 round function was shown in Figure 3.2(a). According to this figure, the round equations are:

$$\begin{aligned} E_i &= D_{i-1} \\ D_i &= C_{i-1} \\ C_i &= ROTL^{30}(B_{i-1}) \\ B_i &= A_{i-1} \\ A_i &= ROTL^5(A_{i-1}) + f_i(B_{i-1}, C_{i-1}, D_{i-1}) + E_{i-1} + K_i + W_i \end{aligned} \tag{3.3}$$

where $+$ is the addition operation performed in *mod* 2^{32}, $ROTL$ denotes the left circular shift operation and $i \in \{1, 2, ..., 80\}$ is the round number. To add a pipeline register in the SHA-1 round, we split the computation of (3.3) into two parts. To double the maximum frequency, the two parts should have the same delay in their critical path. From Figure 3.2(a) and also (3.3), it can be seen that the longest path in the SHA-1 round computation is from E to A which contains four addition operations. Thus, this path should be broken into two identical parts to add a pipeline stage in the round computation.



(a) $A_i = E_{i-1} + \cdots$ (b) $E_{i-1} = A_i - \cdots$

Figure 3.6: Carry-Save-Adder tree for SHA-1 round operation (operands are 32 bits long)

To accelerate the round computation in SHA, a chain of Carry-Save-Adders (CSAs) is usually used along with a Carry-Propagate-Adder (CPA) at the end to perform the addition operation. The five-operand CSA chain for SHA-1 round is shown in Figure 3.6(a). The CSA has a delay of one Full-Adder (FA) regardless of the operands' size. Depending on the type of the CPA used in Figure 3.6(a), we will have different values for the critical path delay. For instance, a n-bit Ripple-Carry-Adder has the propagation delay of $n * delay(FA)$ which would be $32 * delay(FA)$ in this case. Thus, the total delay of the SHA-1 round computation using a Ripple Carry Adder would be $35 * delay(FA)$ (ignoring the delay required for function f_i). Therefore, the pipeline register should be inserted inside the CPA structure. The exact location of the pipeline register in the SHA structure will depend on the the type of Carry-Propagate-Adder used. Ideally, the pipeline stage should be added in a way that the critical path of the SHA round computation be broken into two identical parts which will double the maximum frequency of the design.

The pipeline stage is used for the time redundancy purpose. In time redundancy schemes, each message is processed twice to detect the transient faults. As mentioned before, the inverse of the round computation for the SHA hash functions can be used for this purpose. To do so, the first half round operation shown in Figure 3.5 should be able to switch between the addition and subtraction operations. The addition would be based on the the last part of (3.3). The subtraction should perform inversely to

calculate E_{i-1} as follows:

$$\begin{aligned} E_{i-1} &= A_i - ROTL^5(A_{i-1}) - f(B_{i-1}, C_{i-1}, D_{i-1}) - K_i - W_i \\ &= A_i - ROTL^5(B_i) - f(ROTR^{30}(C_i), D_i, E_i) - K_i - W_i \end{aligned} \tag{3.4}$$

Four negative operands are needed to implement the equation (3.4). Using the two's complement representation of negative numbers, (3.4) would be:

$$\begin{aligned} E_{i-1} &= A_i - ROTL^5(B_i) - f(ROTR^{30}(C_i), D_i, E_i) - K_i - W_i \\ &= A_i + \overline{ROTL5(B_i)} + \overline{f(ROTR^{30}(C_i), D_i, E_i)} + \overline{K_i} + \overline{W_i} + 4 \end{aligned} \tag{3.5}$$

where the $\bar{x}$ is the bitwise complement of the operand x. Using the CSA tree shown in Figure 3.6(a), the computation of (3.5) can be implemented as shown in Figure 3.6(b). Compared to Figure 3.6(a), one more CSA is required in Figure 3.6(b). However, the delay of their critical paths are the same.

The same circuit can be used for the addition and subtraction operations. The inputs to the CSAs would be different for addition and subtraction operations. Therefore, some multiplexors are required to select between the five operands and their counterparts for round and inverse-round computation respectively. The sixth operand should be switched between 0 and 4 for addition and subtraction, respectively. This structure is shown in Figure 3.8. All the CSAs are depicted by a block called CSAs in this figure. From Figure 3.6(a), one can find that the inputs to the CSAs block for SHA-1 round computations are W_i, K_i, E_{i-1}, $ROTL^5(A_{i-1})$ and $f_i(B_{i-1}, C_{i-1}, D_{i-1})$. On the other hand, the inputs to the CSAs block in the inverse-round computation are $\overline{K_i}$, $\overline{W_i}$, A_i, $\overline{ROTL^5(B_i)}$ and $\overline{f_i(ROTR^30(C_i), D_i, E_i)}$. The first five multiplexors select between these values for the round and inverse-round computation. There is also another multiplexor which chooses between 0 and 4. This is because five operands are added in the round computation while the inverse-round computation contains six operands. There are also another three multiplexors which select between A_i, $ROTL^{30}(B_i)$ and C_i in the round computation and $ROTR^30(C_i)$, D_i and E_i in

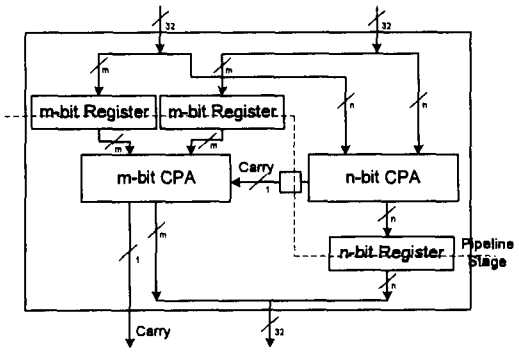


Figure 3.7: 32-bit Pipelined Carry Propagate Adder

the inverse-round computation. A_p, B_p, C_p, D_p, E_p, K_p and W_p are the pipeline registers which store the values of input variables. Before the pipeline stage, the first input of each multiplexor is used in the round computation and the second input is used in the inverse round. This is opposite to the second half of the round computation which is after the pipeline stage. This results from the fact that while the first half of the circuit is performing the round computation, the second half is performing the inverse-round computation and vice versa. The two multiplexors after the pipeline stage select between the CPA output and the A_p and E_p registers. For example, the output value of A would be the CPA output in the round computation ($A_i = ROTL^5(A_{i-1}) + f(B_{i-1}, C_{i-1}, D_{i-1}) + E_{i-1} + K_i + W_i$) and B_i in the inverse-round ($A_{i-1} = B_i$) which comes from the A_p register. The "select" input of all multiplexors (before and after pipeline stage) are connected together. (This input has not been shown in Figure 3.8)

The pipeline stage goes through the Carry-Propagate-Adder as well. The location of the pipeline registers inside the CPA depends on the type of CPA used in the design and also on the implementation limitations. The critical path of SHA-1 round computation contains the CSA tree and the CPA. Considering the delay of the the CSA and the CPA, the pipeline stage should be inserted inside the the CPA. For example, a pipelined 32-bit adder is shown in Figure 3.7. In this figure, a 32-bit adder (which is the case for SHA-1) has been broken into two m-bit and n-bit adders such as $m + n = 32$ and pipeline registers have been added between the two adders.

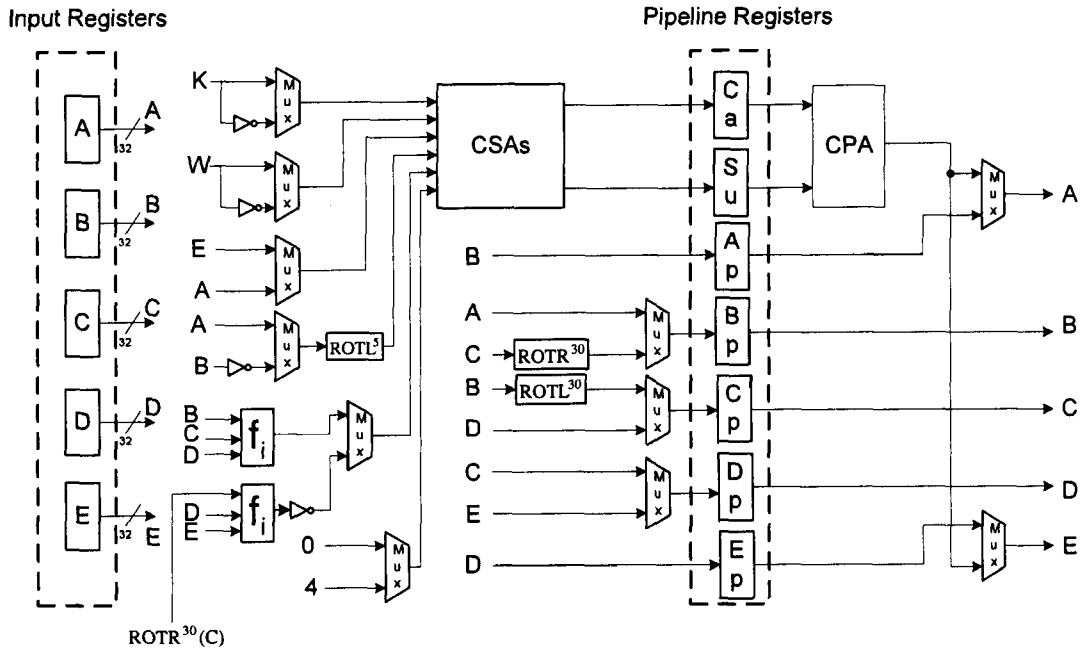
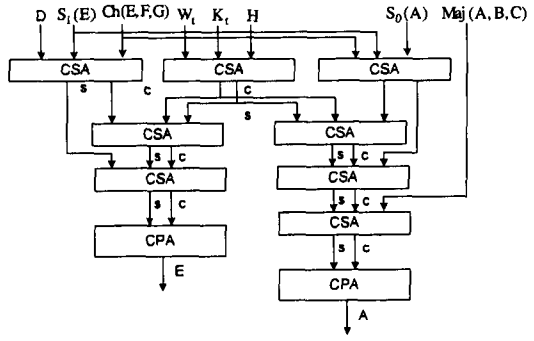


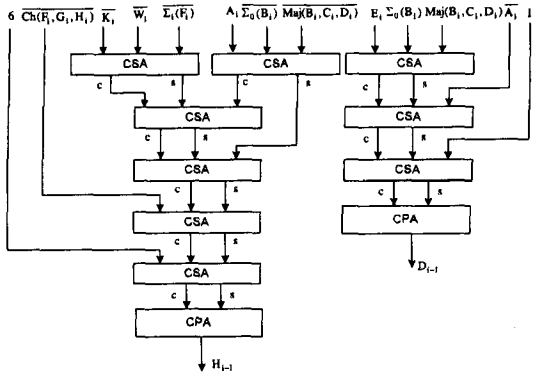
Figure 3.8: Pipelined SHA-1 Round and Inverse-Round Computation

It takes two clock cycles for this adder to add two operands. At the first clock cycle, the n low-significant bits of the two inputs are added by the n-bit CPA. The result of this adder is stored in one n-bit register and a 1-bit carry flip-flop at the second clock cycle. Meanwhile, the m high-significant bits are stored in two m-bit registers added by the m-bit adder using the carry stored in carry register. The output of the n-bit register and m-bit CPA are combined together to constitute the 32-bit output. This structure has been used in Figure 3.8. To speed-up the addition operation, a carry-look-ahead adder has been used as the CPA. Implementing the design on FPGA shows that the values of m and n should be 28 and 4 respectively to have two almost identical delays before and after the pipeline registers.

Two separate pieces hardware have been used in the implementation of function f_i. One for round computation and the other one for inverse round computation. This makes the design capable of detecting the faults inside the function f_i.



(a) addition



(b) subtraction

Figure 3.9: Carry-Save-Adder tree for SHA-512 round computation

3.3.2 SHA-512

The idea introduced in the previous section can also be applied to the SHA-512 to insert a pipeline stage to the round computation for the time redundancy purpose. According to Figure 3.2(b) the round equations for SHA-512 are as follows:

$$A_i = H_{i-1} + \sum_{1}(E_{i-1}) + Ch(E_{i-1}, F_{i-1}, G_{i-1}) + K_i + W_i + \sum_{0}(A_{i-1})$$
$$+ Maj(A_{i-1}, B_{i-1}, C_{i-1})$$
$$E_i = D_{i-1} + H_{i-1} + \sum_{1}(E_{i-1}) + Ch(E_{i-1}, F_{i-1}, G_{i-1}) + K_i + W_i$$
$$B_i = A_{i-1}$$
$$C_i = B_{i-1}$$

$$
\begin{aligned}
D_i &= C_{i-1} \\
F_i &= E_{i-1} \\
G_i &= F_{i-1} \\
H_i &= G_{i-1}
\end{aligned}
\tag{3.6}
$$

where all the operands are 64 bits long and $\sum_0$, $\sum_1$, Ch and Maj are defined in Appendix A. From (3.6), one can find that the longest path in the SHA-512 round computation is from H to A which contains six addition operations. There are also five addition operations from D to E. The structure of the CSA tree for these addition operations is shown in Figure 3.9(a). Seven operands are added in one path and six are added in the other path. The equations for the inverse round computation can be obtained from Figure 3.4(b):

$$
\begin{aligned}
H_{i-1} &= A_i - \sum_0 (A_{i-1}) - Maj(A_{i-1}, B_{i-1}, C_{i-1}) - \sum_1 (E_{i-1}) \\
&\quad - Ch(E_{i-1}, F_{i-1}, G_{i-1}) - K_i - W_i \\
&= A_i - \sum_0 (B_i) - Maj(B_i, C_i, D_i) - \sum_1 (F_i) - Ch(F_i, G_i, H_i) \\
&\quad - K_i - W_i \\
&= A_i + \overline{\sum_0 (B_i)} + \overline{Maj(B_i, C_i, D_i)} + \overline{\sum_1 (F_i)} + \overline{Ch(F_i, G_i, H_i)} \\
&\quad + \overline{K_i} + \overline{W_i} + 6
\end{aligned}
\tag{3.7}
$$

$$
\begin{aligned}
D_{i-1} &= E_i - H_{i-1} - Ch(F_i, G_i, H_i) - \sum_1 (F_i) - K_i - W_i \\
&= E_i - A_i + \sum_0 (B_i) + Maj(B_i, C_i, D_i) \\
&= E_i + \overline{A_i} + \sum_0 (B_i) + Maj(B_i, C_i, D_i) + 1
\end{aligned}
$$

The same idea of Figure 3.6(b) can be applied to Figure 3.9(a) to implement (3.7). This is shown in Figure 3.9(b). According to Figure 3.9, the same circuit with some

modifications could be used to implement the addition and subtraction operations. This circuit is shown in Figure 3.10. Figure 3.10 is similar to Figure 3.8; all the signals and registers in this figure are 64 bits long. The multiplexors are used to select between the inputs of CSA adders, the pipeline registers and the output values in the round and inverse-round operation. Two blocks of CSAs and two blocks of CPA have been used in the design. Like SHA-1, Carry-Look-Ahead adders have been used as the CPA. The idea of Figure 3.7 have been used to implement a 64-bit pipelined adder. The best values for m and n to have two identical critical path delay before and after the pipeline registers are 56 and 8 respectively. Separate hardware have been used to implement $\sum_0$, $\sum_1$, Ch and Maj functions to make the design capable of detecting errors inside these functions.

3.4 Experimental Results

To evaluate the error detection capability of the proposed design, it was simulated for SHA-1 using the programming C language. Then, different cases of single-bit and multiple-bit faults were injected to the design. The faults were injected at the input, adders' outputs, functions' outputs and registers' outputs in the design. Stuck-at faults (both stuck-at-0 and stuck-at-1) were inserted as single-bit and multiple-bit faults in the design. For each fault case, the round computation was tested using 1000000 different random inputs. In all of these cases, the error was detected by the design and 100% error coverage was achieved.

Adders are the most important elements of SHA-1 and SHA-512 round computations. The time redundancy method proposed in this paper was designed to detect errors which occur in the addition operation. Any fault in the adders would be detected using this design. Besides adders, there are some other parts in the SHA-1 and SHA-512 round computation. SHA-1 contains function f while SHA-512 has $\sum_0$, $\sum_1$, Ch and Maj functions. The inputs to these functions in the round and inverse-round computations are the same. Thus if we use the same hardware to implement these functions, the errors inside these functions would be undetected. To overcome this

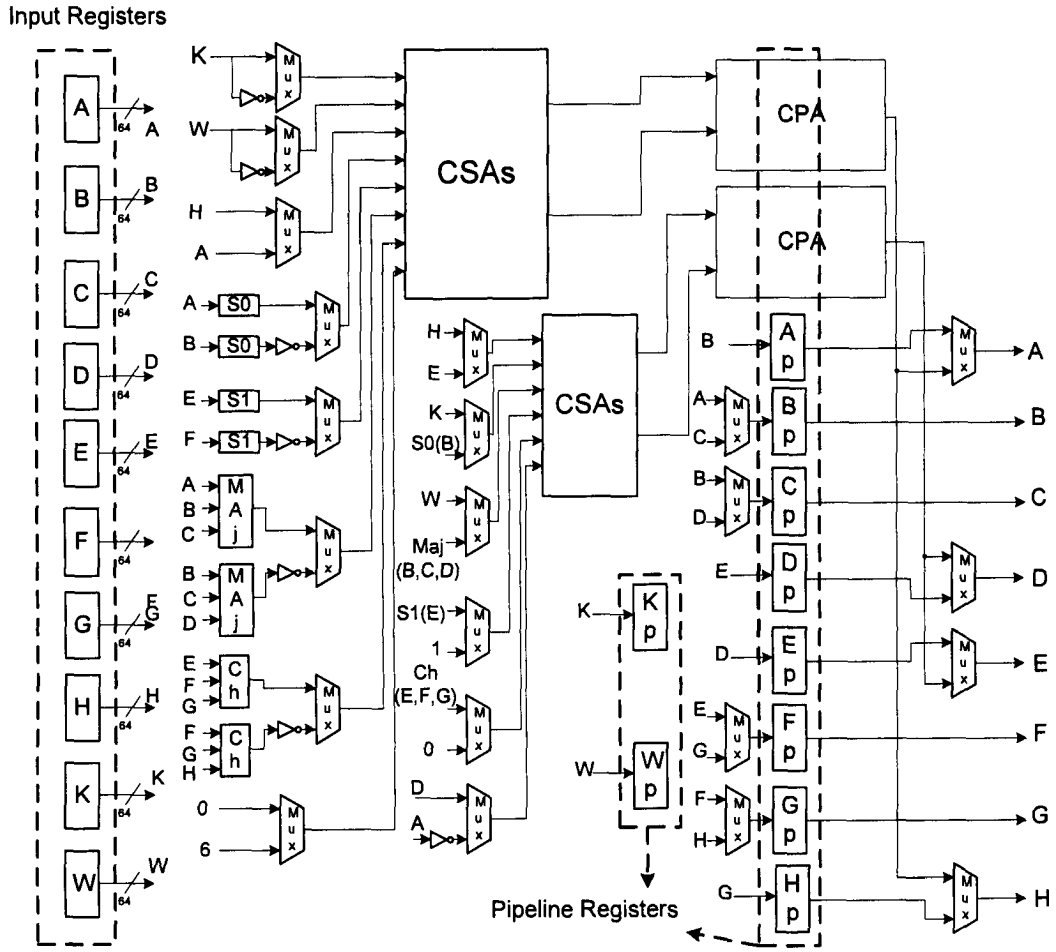


Figure 3.10: Pipelined SHA-512 Round and Inverse-Round Computation

problem, we used hardware redundancy to implement these functions. Two separate pieces of hardware were used to implement each function, one for round computation and one for the inverse-round computation. This hardware is shown in Figure 3.8 and Figure 3.10. This makes the proposed design capable of detecting the faults inside these functions as well as other parts of the design. Because of the simplicity of these functions compared to the entire round computation, this hardware redundancy technique does not impose significant overhead to the proposed structure.

The proposed designs for SHA-1 and SHA-512 were implemented on FPGA to verify the results discussed in the previous section. The target platform was a Xilinx

Table 3.2: Implementation results of SHA-1 on xcv2p7

SHA-1 round Computation	Frequency (MHz)	Throughput (Mbit/s)	Number of Slices
Without Error Detection	66	422	341
With Error Detection	130	408	539

Table 3.3: Implementation results of SHA-512 on xcv2p7

SHA-512 round Computation	Frequency (MHz)	Throughput (Mbit/s)	Number of Slices
Without Error Detection	41	524	1584
With Error Detection	76	477	2062

VirtexII Pro FPGA. Two structures were implemented for each of SHA-1 and SHA-512 round computations. The first structure is without error detection and the second implementation is with the fault detection for SHA-1 and SHA-512 round computations. The message padding and the message scheduler for both implementations are the same and the fault detection method is only applied to the round computation. To compare the implementation results of two structures, the round computation for both implementations was synthesized, mapped, placed and routed separately. Tables 3.2 and 3.3 show the results of these two implementations for SHA-1 and SHA-512 respectively.

As seen from Tables 3.2 and 3.3, the maximum frequency of the implementations without error detection is slightly less than twice the frequency of first implementation. This is because of the fact that the critical path of the SHA-1 and SHA-512 implementations cannot be split into two parts with exactly the same delay in practice. The multiplexors added to the second implementation are another reason for this difference. The throughput is obtained from the following equation:

$$Throughput = \frac{blocksize \times frequency}{\#clockcycles}. \tag{3.8}$$

In SHA-1, the messages are processed in 512-bit blocks. Thus, the number of bits in SHA-1 would be 512. The number of clock cycles for the implementation without fault detection is 80 while it takes 163 clock cycles to process each block of data in the implementation with fault detection. Table 3.2 shows that there is a 3 percent degradation in the throughput of the proposed design for SHA-1 and also a 58 percent

Table 3.4: Comparison of results for fault detection methods in SHA-512

Fault Detection Design	Throughput Overhead	Area Overhead
[15]	11.6%	21%
[16] (HCMainRegs)	73%	100%
proposed	10%	30%

increase in the number of slices used compared to the original implementation of the SHA-1 hash function. Similarly, Table 3.3 shows that the SHA-512 implementation with fault detection takes 10 percent more time and requires 30 percent more resources compared to the original implementation.

Table 3.4 shows the timing and area overhead comparison of the proposed design for SHA-512 with two similar works. The work presented in [15] has used parity bits to detect faults in SHA-512. The timing overhead of this design is almost the same as our proposed design while its area overhead is less than ours. But because of using parity bits for fault detection, the design proposed in [15] is only capable of detecting faults with odd number of erroneous bits while our proposed design is able to detect any kind of faults and errors. Table 3.4 also shows the results of the work presented in [16]. In this work, several fault detection method for SHA-2 hash family has been proposed best of which has been shown in Table 3.4 for SHA-512 hash function. This table indicates that our design has much better performance in terms of timing and area overhead compared to [16].

3.5 Conclusion

A fault detection scheme based on time redundancy and pipelining has been proposed for the hardware implementations of the SHA-1 and SHA-512 round computations. Because the SHA hash functions mainly include addition operations in their critical path, the subtraction operation has been used as the inverse function for the redundancy purpose. This makes our scheme capable of detecting permanent as well as temporary faults as opposed to normal time redundancy techniques which are only capable of detecting transient faults. Because of using the pipelining method, this time redundancy technique does not add significant timing overhead to the original

design. The FPGA implementations of the proposed designs for SHA-1 and SHA-512 show that there is a low overhead in the throughput and the area utilization of the this scheme as compared to the traditional double-modular redundancy-based scheme. The timing overhead of this design is less than the parity-based redundancy schemes.

Bibliography

[1] W. Stallings, *Cryptography and network security*, Prentice Hall Upper Saddle River, NJ, 2003.

[2] National Institute of Standards and Technology, "Secure Hash Standard (SHS), FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION, FIPS PUB 180-3", (2008).

[3] RFC 2401, "The Security Architecture for the Internet Protocol", (1998).

[4] L. Dadda, M. Macchetti and J. Owen, "The design of a high speed ASIC unit for the hash function SHA-256 (384, 512)," *Proc. Design, Automation and Test in Europe Conference and Exhibition*, 2004.

[5] T. Grembowski, R. Lien, K. Gaj, N. Nguyen, P. Bellows, J. Flidr, T. Lehman and B. Schott, "Comparative Analysis of the Hardware Implementations of Hash Functions SHA-1 and SHA-512," *Proc. 5th International Conference on Information Security, ISC*, 2002, pp. 75-89.

[6] M. Macchetti and L. Dadda, "Quasi-pipelined hash circuits," *Proc. IEEE Symposium on Computer Arithmetic*, 2005, pp. 222-229.

[7] F. Crowe, A. Daly, T. Kerins and W. Marnane, "Single-chip FPGA implementation of a cryptographic co-processor," *Proc. IEEE International Conference on Field-Programmable Technology*, pp. 279-285, 2004.

[8] R. Lien, T. Grembowski, and K. Gaj, "A 1 Gbit/s Partially Unrolled Architecture of Hash Functions SHA-1 and SHA-512," *Springer J. CT-RSA*, pp. 324-338, 2004.

[9] N. Sklavos, G. Dimitroulakos and O. Koufopavlou, "An ultra high speed architecture for VLSI implementation of hash functions," *Proc. 10th IEEE International Conference on Electronics, Circuits and Systems, ICECS,* 2003, vol. 3, pp. 990-993.

[10] A.P. Kakarountas, G. Theodoridis, T. Laopoulos and C.E. Goutis, "High-Speed FPGA Implementation of the SHA-1 Hash Function," *Proc. IEEE Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS,* 2005, pp. 211-215.

[11] R. Chaves, G. Kuzmanov, L. Sousa and S. Vassiliadis, "Cost-Efficient SHA Hardware Accelerators," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems,* vol.16, no.8, pp.999-1008, 2008.

[12] H. Michail, A.P. Kakarountas, O. Koufopavlou and C.E. Goutis, C.E., "A low-power and high-throughput implementation of the SHA-1 hash function," *Proc. IEEE International Symposium on Circuits and Systems, ISCAS* 2005. Vol. 4, pp. 4086-4089.

[13] G. Wang, "An Efficient Implementation of SHA-1 Hash Function," *Proc. IEEE International Conference on Electro/information Technology,* 2006, pp.575-579.

[14] H. Bar, H. Choukri, D. Naccache, M. Tunstall and C. Whelan, "The Sorcerers Apprentice Guide to Fault Attacks," *Proc. IEEE* , vol. 94, no. 2, pp. 370-382, 2006.

[15] I. Ahmad, and A.S. Das, "Analysis and Detection Of Errors In Implementation Of SHA-512 Algorithms On FPGAs," *The Computer Journal,* vol.50, no.6, pp.728-828, 2007)

[16] M. Juliato, C. Gebotys and R. Elbaz, "Efficient fault tolerant SHA-2 hash functions for space applications," *Proc. IEEE Aerospace conference,* 2009, pp.1-16.

[17] G. Bertoni, L. Breveglieri, I. Koren and P. Maistri, "An efficient hardware-based fault diagnosis scheme for AES: performances and cost," *Proc. 19th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, DFT'04,* 2004, pp. 130-138.

[18] C.H. Yen and B.F. Wu, "Simple Error Detection Methods for Hardware Implementation of Advanced Encryption Standard," *IEEE Transaction on Computers,* vol. 55, pp. 720-731, 2006.

[19] K. Wu, R. Karri, G. Kuznetsov and M. Goessel, "Low Cost Concurrent Error Detection for the Advanced Encryption Standard," *Proc. International Test Conference on International Test Conference*, 2004, pp 427-435.

Chapter 4

Implementation of SHA Hash Functions on Wireless Sensor Boards

4.1 Introduction

There has been an increasing growth in the use of WSNs (wireless sensor networks). A wireless sensor network consists of a large number of distributed nodes called sensor nodes which monitor the physical conditions such as temperature, pressure, humidity, vehicular movements etc. Sensor nodes consist of sensing, data processing and communication components. Because of their low cost and low power structure, hundreds of sensor nodes can be deployed for a special purpose to make a wireless sensor network. There are so many potential applications for wireless sensor networks. They can be used in different application areas such as military, industry, home, agriculture etc. In military applications, they are used to detect special kinds of weapons such as nuclear and chemical weapons. They can be used in commercial buildings to monitor and control their temperature, light, alarms, etc. Agriculture applications of wireless sensor networks include the monitoring of the physical conditions such as temperature and soil conditions.

Industrial environments are one of the main application areas of wireless sensor networks. They can be used for monitoring and surveillance of different components of an industrial environment. For example, [1] describes the use of WSNs in monitoring oil pipelines near the Arctic Circle. The temperature of the pipes should be controlled to prevent them from bursting. The pipes need to be heated if their temperature gets too low. For this purpose, the wired sensors which cost thousands of dollars to install were replaced by sensor nodes which are less expensive and proved to have more reliable reading performance compared to the wired sensors. They can also be used for inventory management purposes in chemical plants. They enable instant access to the real-time tank inventory data which makes it easier to manage and schedule them to have a constant supply of raw materials [2]. Pulp and paper mills are another industrial application of wireless sensor networks. They are attached to the rolling machines of pulp and paper mills to monitor the temperature, speed, pressure or vibration of different parts of the machine or the process line. Wireless sensor networks can also be used as a heat tracing solution in oil refinery systems. A large number of sensor nodes can be distributed in the pipes which makes it possible to have global monitoring as opposed to the local wired monitoring systems. Nuclear power plants are another area which can use wireless sensor networks for radiation monitoring purposes.

As can be seen from the above examples, there are so many potential areas in which WSNs can be utilized to improve performance and reduce the cost of the system compared to traditional wired sensor networks. There are two main features which make wireless sensor networks suitable for various applications. First, they are small in size which enables them to be deployed in different environments and secondly, their wireless communication makes their installation easier and more cost-efficient. Therefore, the cost of a sensor node and its installation is much less than a typical wired sensor and large numbers of them can be deployed for a single application.

Because of the wireless communication in sensor networks, They are vulnerable to security attacks. Therefore, in almost every application area, wireless sensor networks require security measures to be resistent to security attacks. There are several kinds

of attacks that can threaten a wireless sensor network:

- **Denial-of-Service Attacks**: These kinds of attacks threaten the service availability of the system. This can be done by jamming the signals at the physical layer of the network or by flooding messages at the data link layer. The specifications of the MAC protocol used by the wireless sensor networks can be exploited to perform DoS attacks. DoS attacks could result in the waste of power which is a major issue in sensor nodes.

- **Eavesdropping**: This kind of attack threatens the confidentiality of the messages transmitted by the sensor nodes. It is usually performed at the application layer of the network where the application data exist. Encryption and decryption techniques are the best countermeasures for eavesdropping attacks.

- **Message Modification**: This is one of the main types of attacks in wireless sensor networks. These types of attacks threaten the integrity of transmitted messages. They aim to modify the messages transmitted through the network which might result in major damage especially in critical applications. Changing the values read by the sensor or corrupting the routing information in the packets could drastically affect the performance of the network.

The attacks mentioned above are the main types of attacks in WSNs. There are some other types of attacks which can be considered as one of the above categories. For example, creating false packets and sending them to the network which threatens the authenticity of the messages can be categorized as a message modification attacks. Message authentication and integrity protection are closely related in security terminology.

These attacks indicate the vulnerability and the need for security in wireless sensor networks. The security issues in wireless senor networks has been recently studied in literature [4], [5], [6]. Because of the hardware constraints existing in sensor nodes, it is usually very challenging to apply security mechanisms to wireless sensor networks. Because of their structure, sensor nodes usually suffer from limited computational

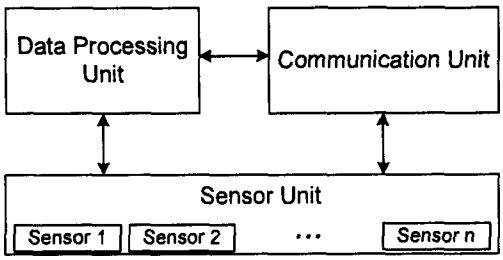


Figure 4.1: The Structure of Wireless Sensor Boards

power, limited memory and limited power supply. On the other hand, cryptographic algorithms which are used in security mechanisms are usually complex algorithms which require a significant amount of memory and processing power. This makes it quite challenging to implement cryptography algorithms on wireless sensor boards.

There has been some research on the implementation of cryptographic algorithms on sensor boards. In [3], TinyEcc has been introduced which is the implementation of Elliptic Curve Cryptography (ECC) on senor boards. The SHA-1 and HMAC hash functions have been implemented on sensor boards in [7]. In this paper, we implement the SHA-1 and SHA-512 hash functions on sensor boards and compare their implementations in terms of memory requirements and execution time. This comparison will be helpful in choosing these hash functions for a specific applications. Hash functions are used for integrity protection and message authentication which are crucial in industrial applications in which the integrity of data transmitted is the major security issue.

The rest of this paper is organized as follows: In Section 2, the structure of sensor boards and the comparison of different sensor boards available will be presented. Section 3 introduces the TinyOS and the NesC language used in programming the sensor boards. The architecture of SHA-1 and SHA-512 is presented in Section 4. In Section 5, the results of SHA-1 and SHA-512 implementations on a Micaz sensor board is presented and finally the paper concludes in Section 5.

Table 4.1: Comparison of Crossbow Sensor Boards

Sensor Board	Microcontroller	frequency Band	Flash Memory	RAM
TelosB	TI MSP430	2.4-2.48GHz	48Kb	10Kb
Micaz	MPR2400(Atmega128L)	2.4-2.48GHz	128Kb	4Kb
Iris	XM2110CA(Atmega128L)	2.4-2.48GHz	128Kb	8Kb

4.2 Structure of Wireless Sensor Networks

There are three parts in a sensor node. As shown in Figure 4.1, a sensor node consists of the Sensor unit, the Data Processing unit and the Communication unit. The Sensor unit is the collection of different sensors to measure temperature, pressure, etc. There are many kinds of sensors which can be used for various applications. The Data processing unit is a microcontroller which stores and processes the data read by sensors and the data used for the communication of sensor nodes. The microcontroller usually has an 8-bit or 16-bit RISC core and consists of RAM, flash memory and/or EEPROM and operates at a frequency of a few MHz. The Communication unit is the wireless transceiver used for the transmission and receiving of the data by the sensor nodes. The technology used in this part is usually an IEEE 802.15.4/Zigbee-compliant system. Table 4.1 shows a comparison of three famous sensor boards provided by Crossbow Technology: Micaz, TelosB and Iris. This table shows the type of microcontroller used in the board and the frequency, RAM, Flash Memory available on them.

4.3 TinyOS and NesC Language

TinyOS [8] is an open-source operating system designed for sensor boards. The philosophy behind TinyOS was to provide a framework for programming in embedded systems which requires the code size and the execution time to be minimized. It consists of a set of components which are used to develop custom applications on sensor boards. Its component library includes the network protocols, sensor drivers and data acquisition tools which can be refined for custom applications.

TinyOS has a component-based and event-driven programming model. The components are organized into layers; The lower the layer, it is closer to hardware compo-

nents and vice versa. A TinyOS application is a collection of components connected together. There are three computational concepts in TinyOS: Commands, Events and Tasks. Commands and events are exchanged between components while tasks are executed inside the components. Components use commands to ask for a service from other components. The completion of a service requested by a command is signalled by events. For example, a component can request a timer component to start a 1ms timer. This can be done by a command. The timer component would issue a timer event each 1 milliseconds. Hardware interrupts are another kind of event which can be signaled asynchronously. Tasks are functions executed inside the components and they only access the information within a component. The commands and events may return immediately while deferring the extensive computation to tasks. There are two threads of execution in a TinyOS application. one for tasks and one for the hardware event handlers. Hardware event handlers are executed in response to hardware interrupts. Their execution may preempt the execution of a task or other event handlers.

A TinyOS application is a set of components wired together to implement the required service. The TinyOS applications and libraries are written in a component-based language called NesC [9]. The NesC language is primarily intended for embedded systems. It has a C-like syntax which supports the implementation and linking of the components in the TinyOS environment. It is actually an extension to C designed to support the component-based and event-driven architecture of TinyOS. There are some basic concepts in the NesC language:

Application: An application is a set of components linked together to perform a required service.

Component: Components are the building blocks of a TinyOS application written in NesC. Components are linked together via interfaces. Through the interfaces, the components can send commands and receive events from other components.

Interface: An interface is used to connect the components to each other. It specifies a set of functions as the interface's commands and a set of functions as their events. The interfaces are bidirectional. For a component to call a command of an interface,

it must implement the event of that interface. The interfaces are the only point of access to the components. A component can have multiple interfaces.

Module: A module is the implementation of one or more interfaces. Modules are components that provide interfaces and implement the commands of the interfaces they provide. The implementation of commands in NesC is similar to the implementation of function bodies in C.

Configuration: A configuration is a component that wires the other components together. Wiring is performed by connecting the interfaces provided by a component (module) to the interfaces used by other components. The relation between a module and a configuration in NesC is similar to the relation between function definition and function declaration in C. A configuration specifies the connection between the components through interfaces while a module implements the functions provided by one or more interface.

Based on the above concepts, each TinyOS application should contain a configuration and a module. The configuration specifies the components that the application uses and the module implements the commands that the application component provides. Similarly, NesC libraries contain interfaces and their modules which are used by the applications.

4.4 The SHA Hash Functions

The SHA-1 hash function generates a 160-bit output from the arbitrary length input message. The general structure of the SHA-1 hash value computation is shown in Figure 4.2. It contains four main units: The message padding unit, the block expansion unit, the round computation and the final hash computation unit. The input message is split and padded into blocks of 512 bits. The block expansion unit generates 80 × 32-bit W_is from each block, one 32-bit for each round. The SHA-1 round equations are as follows:

$$E_i = D_{i-1}$$

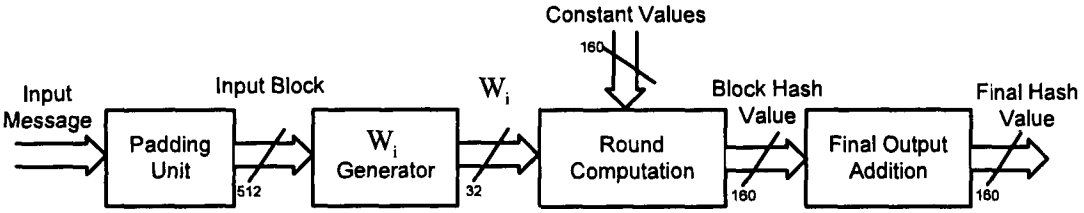


Figure 4.2: SHA-1 Architecture

$$
\begin{aligned}
D_i &= C_{i-1} \\
C_i &= ROTL^{30}(B_{i-1}) \\
B_i &= A_{i-1} \\
A_i &= ROTL^{5}(A_{i-1}) + f_i(B_{i-1}, C_{i-1}, D_{i-1}) + E_{i-1} + K_i + W_i
\end{aligned}
\tag{4.1}
$$

where A, B, C, D and E are 32-bit operands, $+$ is $mod\, 2^{32}$ addition and $ROTL$ denotes the left circular shift operation. From (4.1), one can see that the round computation of the SHA-1 is comprised of the addition, logical rotation and the function f_i. The round computation will be performed 80 times with a constant initial value for A, B, C, D and E. The output of the last round will be added to the output of the other blocks to generate the final hash value. There are four different functions f_i and four constants K_i, each for every 20 rounds which are defined in Appendix A. The 80 W_is for each block of the input message are obtained from:

$$
W_i = \begin{cases} M_i & 0 \le i < 16 \\ RotL^1(W_{i-3} \bigoplus W_{i-8} \bigoplus W_{i-14} \bigoplus W_{i-16}) & 16 \le i < 80 \end{cases}
\tag{4.2}
$$

where M_i denotes the ith 32-bit word of the input data block, $RotL^1$ is the one-bit circular shift to the left and $\oplus$ is the bit-wise XOR operation.

The SHA-512 is one of the three versions of the SHA-2 hash functions which generates a 512-bit hash value. The general structure of SHA-512 is the same as SHA-1 shown in Figure 4.2. The data sizes and the internal structure of units are different from SHA-1. The data blocks are 1024 (16×64) bits long and the the block expansion generates 64-bit W_is which are fed into the round computation. The round

computation equations have been shown in (4.3):

$$
\begin{aligned}
A_i &= H_{i-1} + \sum_1 (E_{i-1}) + Ch(E_{i-1}, F_{i-1}, G_{i-1}) + K_i + W_i + \sum_0 (A_{i-1}) \\
&\quad + Maj(A_{i-1}, B_{i-1}, C_{i-1}) \\
E_i &= D_{i-1} + H_{i-1} + \sum_1 (E_{i-1}) + Ch(E_{i-1}, F_{i-1}, G_{i-1}) + K_i + W_i \\
B_i &= A_{i-1} \\
C_i &= B_{i-1} \\
D_i &= C_{i-1} \\
F_i &= E_{i-1} \\
G_i &= F_{i-1} \\
H_i &= G_{i-1}
\end{aligned}
\tag{4.3}
$$

where all the operands are 64 bits long and $\sum_0$, $\sum_1$, Ch and Maj have been defined in Appendix A. Like the SHA-1 hash function, the round computation for SHA-512 mainly comprises of addition operations which are performed in modulo 2^{64} arithmetic instead of modulo 2^{32} in SHA-1. Similarly, the A,B,C,...H and the K_i constants are 64-bit words. The round computation is performed 80 times in SHA-512 and the 64-bit W_is are obtained from:

$$
W_i = \begin{cases} M_i & 0 \leq i < 16 \\ \sigma_1(W_{i-2}) + W_{i-7} + \sigma_0(W_{i-15}) + W_{i-16} & 16 \leq i < 80 \end{cases}
\tag{4.4}
$$

The + denotes the arithmetic addition operation performed in modulo 2^{64} and σ_0 and σ_1 functions have beenare defined in Appendix A.

4.5 Implementation of SHA Hash Functions on Wireless Sensor Boards

In this section, we describe the implementation of SHA-1 and SHA-512 hash functions on a sensor board. The SHA-512 implementation will be described in detail. The SHA-1 has a similar implementation.

As mentioned in the previous chapter, the Secure Hash Algorithm has four main units: message padding, block expansion, round computation and final hash computation. Combining the block expansion and round computation units, these four units have been implemented with three different functions. The SHA-512 hash function has been implemented as an interface which has three commands called: *PadMessage*, *ComputeRound* and *ComputeFinalHash*:

```
interface SHA512
{
    Command PadMessage(message);
    Command ComputeRound();
    Command ComputeFinalHash();
}
```

This interface should have a module to implement these three commands. This module is called **SHA512M** which contains the implementation of the units mentioned above:

```
module SHA512M
{
    provides interface SHA512;
}
implementation
{
    Command PadMessage(message)
```

```
    {
    }
    Command ComputeRound()
    {
    }
    Command ComputeFinalHash()
    {
    }
        .
        .
        .
}
```

There are some other functions in the implementation which are internal functions of the module and they cannot be accessed from the other components. If a component wants to find the hash value of a message it should be wired to the module **SHA512M** through the interface **SHA512**. By calling the above three commands respectively, the **SHA512** interface calls their implementation to compute the hash value of the input message.

To use and test the **SHA512** interface we have implemented, we need to use another component which uses this interface and calls its commands. For this purpose, we implemented a component called testSHA512 to test the SHA512 interface and measure its execution time. This component is used as the top component which is compiled and downloaded on the sensor board. It uses the SHA512 interface and calls its commands. It also uses some other interfaces which are used for testing and measuring the execution time. These interfaces are: **Leds** which controls the LEDs available on the sensor board and is used for debugging purposes and **SysTime** which is used to measure the execution time of the SHA512 implementation. To debug the implementation, the SHA512 commands are called in the body of the testSHA512 module. A message is provided and the commands are called respectively to pad and compute the hash value of the message. To measure the execution time

Table 4.2: Implementation results SHA-1 and SHA-512 on Micaz sensor board

Hash Function	Throughput(kbit/s)	ROM-RAM(bytes)
SHA-1	60	3300-207
SHA-512	30	20854-991

Table 4.3: Implementation results SHA-1 and SHA-512 on Micaz sensor board

SHA-1 Implementation	Time(ms)	ROM-RAM(bytes)
[7]	35	3504-140
ours	8.6	3300-207

of this procedure, the current time of the system is measured before and after the SHA512's command calls and their difference is computed. This is done using the SysTime interface which provides commands to get the current time of the systems. The commands are called by the testSHA512 module which uses the SHA512 as an interface. The PseudoCode below shows the procedure of measuring the execution time:

```
t1 = call SysTime.getTime32();
call SHA512.SHA512PadMessage(message);
call SHA512.ComputeRound();
call SHA512.ComputeFinalHash();
t2 = call SysTime.getTime32();
```

$t_2 - t_1$ is the execution time of the SHA512 implementation. This code is used to measure the hash function computation time for input messages with different sizes. The same method has been used to test and measure the execution time of the SHA-1 implementation. Like SHA-512, the SHA-1 has been implemented as an interface and used by a top module for testing and debugging purposes.

The SHA-1 and SHA-512 were implemented on the Micaz sensor board. The results of the implementations are shown in Table 4.2. This table shows that there are 3300(207) bytes of ROM(RAM) required for the implementation of the SHA-1 hash function and it takes 20854(991) bytes of ROM(RAM) to implement the SHA-512 on the Micaz sensor board. It also shows that it takes 8.6 milliseconds to compute the hash value of a single block in SHA-1 resulting in a 60 kbits/sec throughput. The throughput for SHA-512 is 30 kbits/sec.

To the best of our knowledge, the only work that has considered the implementation of hash functions on sensor boards is [7]. In [7] the SHA-1 and HMAC hash functions have been implemented on TelosB sensor board. The comparison results of our SHA-1 implementation and [7] are shown in Table 4.3. This table shows that our implementation has better performance in terms of execution time and ROM required while spends more RAM compared to [7]. There has been no published implementation of SHA-512 on wireless sensor boards.

4.6 Conclusion

Wireless Sensor Networks are going to be used in various applications. They are used in industrial environments for configuration and monitoring purposes. Because of the wireless communication of the nodes, security is an important issue in such networks. Because of the resource limitations in terms of processing power, memory and power consumption, implementation of cryptographic algorithms on sensor boards is quite challenging. In this paper, we implemented the SHA-1 and SHA-512 hash functions on sensor boards to measure the resource and timing requirements of these functions and consider their suitability for different applications. For this purpose, we implemented the SHA-1 and SHA-512 hash functions using NesC language in TinyOS and downloaded it on Micaz sensor board. We measured the ROM, RAM required and also the the execution time of the implementations. The results show that our SHA-1 implementation has a better execution time and ROM required compared to the previous work.

Bibliography

[1] N. Goh, "Wireless Sensor Networks Ubiquitous Watchdogs of the Future?," INTRO Newsletter, August 2003. URL http://www.nus.edu.sg/intro/newsletter0311.shtml.

[2] X. Shen, Z. Wang, Y. Sun, "Wireless sensor networks for industrial applications," *Proc. Fifth World Congress on Intelligent Control and Automation, WCICA 2004.*, 2004, vol.4, pp. 3636-3640.

[3] A. Liu, P. Ning, "TinyECC: A Configurable Library for Elliptic Curve Cryptography in Wireless Sensor Networks," *Proc. International Conference on Information Processing in Sensor Networks,IPSN '08*, 2008, pp.245-256.

[4] P. Traynor, R. Kumar, H. Choi, G. Cao, S. Zhu, and T. F. L. Porta, "Efficient hybrid security mechanisms for heterogeneous sensor networks," *IEEE Transaction on Mobile Computing,*, 2007 vol. 6, no. 6, pp.663-677.

[5] K. Lu, Y. Qian, M. Guizani, and H.-H. Chen, "A framework for a distributed key management scheme in heterogeneous wireless sensor networks," *IEEE Transactions on Wireless Communications*, 2008, vol. 7, no. 2, pp. 639-647.

[6] R. Azarderakhsh, A. Reyhani-Masoleh, and Z.-E. Abid, "A key management scheme for cluster based wireless sensor networks," *IEEE/IFIP International Conference on Embedded and Ubiquitous Computing"*, 2008, vol.2, pp. 222-227.

[7] H. Li, Y. Choi, H. Kim, "Implementation of TinyHash based on Hash Algorithm for Sensor Network," *Proc. World Academy Of Science, Engineering and Technology,*, 2005, vol. 10, pp. 135-139.

[8] http://www.tinyos.net/

[9] D. Gay, D. Culler and P. Levis, "nesC Language Reference Manual", available at http://webs.cs.berkeley.edu/tos/api/nesc/doc/ref.pdf, 2002

Chapter 5

Conclusions

The widespread use of communication networks in industrial environments has led to new issues in these systems. Network-based control systems are used in various industries. Modern industrial systems have a hierarchical structure in which devices at the different levels from management to field level communicate with each other. Security is one of the issues that has become important in new industrial systems. Traditional industrial networks were stand-alone networks with no or limited connection to outside networks. This is not true for modern control systems in which devices are connected together with standard communication networks. Security issues in industrial networks has been the focus of this thesis.

The focus of this thesis is to implement cryptographic algorithms to be used in industrial environments. Because the data integrity is the major issue in many industrial networks, we implemented hash functions which are one of the main methods of providing data integrity. Industrial devices are usually embedded devices with limited resources such as processing power and memory requirements. We focused on the reliable implementation of SHA hash functions on FPGAs and also wireless sensor boards which are used in embedded systems for various industrial applications. Reliable implementation is needed in critical applications in which any fault and error in the devices might result in grave aftermaths.

We propose a fault detection method for the FPGA implementation of SHA-1 and SHA-512 hash functions. This method which is based on time redundancy is capable

of detecting permanent as well as transient faults. The goal of this implementation is to optimize the design in terms of the delay and area overhead that this time redundancy method imposes on the design. The pipelining technique is used to compensate for the the delay overhead that time redundancy induces to the design. Using this technique we have decreased the delay overhead to as low as 3% for SHA-1 and 10% for SHA-2 while having acceptable area overhead.

Our goal in the sensor board implementation is to optimize the design in terms of speed and ROM and RAM usage. We also aimed to have a comparison between SHA-1 and SHA-512 implementations which is helpful in deciding between these two hash functions in a specific application. Because of the minimal structure of the TinyOS 1.1 and the programming environments, debugging the implementations has been quite a challenge in sensor boards. These limitations in debugging tools have been overcome in TinyOS 2.

5.1 Contributions

This section briefly describes the contributions of this thesis:

- A new fault detection scheme for the FPGA implementation of SHA-1 and SHA-512 has been introduced in this thesis. The main feature of this scheme is that although it is based on time redundancy, it is capable of detecting permanent faults as well as transient faults.

- To the best of our knowledge, this is the first FPGA implementation of the SHA-1 hash function which has the fault detection capability. Because of using the pipelining method, the proposed design has a very small (3%) timing overhead.

- There is an improvement in the timing overhead of the proposed scheme for SHA-512 compared to the previous works. The timing overhead of the proposed design is 10 percent while two previous works show 11.6 percent and 73 percent timing overhead.

- This fault detection scheme is capable of detecting any kind of faults as opposed to the parity-based schemes which only detect faults with an odd number of erroneous bits.
- The sensor board implementation of SHA-1 done in this thesis shows execution time and ROM usage improvement compared to the previous work.
- To the best of our knowledge, this is the only implementation of SHA-512 on sensor boards.

5.2 Future Work

In this thesis, the hardware implementations of SHA-1 and SHA-2 provided by NIST has been studied. NIST has decided to develop one or more additional hash functions through a public competition, similar to the development process of the Advanced Encryption Standard (AES). To be selected as the final SHA-3 hash function, the candidates have to go through a competition in terms of security and implementation aspects. The final hash function will be announced in 2012. Like the AES, implementation issues are one of the major criteria for deciding the final hash function. Implementing the candidates in software and hardware (FPGAs, ASIC, sensor boards etc.) is of great importance. Whether or not the proposed fault detection method can be applied to the SHA-3 candidates is one of the future works.

The proposed fault detection method has been applied to the round computation of the SHA hash functions. It can be extended to the W_i generator unit of SHA-512 which contains addition operation as well. (The W_i generator unit in SHA-1 does not contain addition, therefore this method is not suitable for it.)

Another future direction would be applying other fault detection methods such as hardware redundancy and information redundancy for the implementation of SHA hash functions.

Appendix A

Functions and Constants used in SHA-1 and SHA-512

A.0.1 SHA-1

$$f_i(x,y,z) = \begin{cases} (x \wedge y) \oplus (\overline{x} \wedge z) & 0 \leq i < 20, \\ x \oplus y \oplus z & 20 \leq i < 40, \\ (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z) & 40 \leq i < 60, \\ x \oplus y \oplus z & 60 \leq i < 80. \end{cases}$$

$$K_i = \begin{cases} 5a827999 & 0 \leq i < 20, \\ 6ed9eba1 & 20 \leq i < 40, \\ 8f1bbcdc & 40 \leq i < 60, \\ ca62c1d6 & 60 \leq i < 80. \end{cases}$$

A.0.2 SHA-512

$$Ch(x,y,z) = (x \wedge y) \oplus (\bar{x} \wedge z)$$

$$Maj(x, y, z) = (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z)$$

$$\sum_0 (x) = ROTR^{28}(x) \oplus ROTR^{34}(x) \oplus ROTR^{39}(x)$$

$$\sum_1 (x) = ROTR^{14}(x) \oplus ROTR^{18}(x) \oplus ROTR^{41}(x)$$

$$\sigma_0(x) = ROTR^{1}(x) \oplus ROTR^{8}(x) \oplus SHR^{7}(x)$$

$$\sigma_1(x) = ROTR^{19}(x) \oplus ROTR^{61}(x) \oplus SHR^{6}(x)$$