7-30-2009

Telemedicine patient identification with RFID; an embedded approach

Onder Aksoy

Follow this and additional works at: http://scholarworks.rit.edu/theses

Recommended Citation

Telemedicine Patient Identification with RFID;
An Embedded Approach

by

Onder Aksoy

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Science in Computer Engineering

Supervised by

Dr. Andreas Savakis
Department of Computer Engineering
Kate Gleason College of Engineering
Rochester Institute of Technology
Rochester, NY
July 30, 2009

Approved By:

_____________________________________________        ___________      ___
Dr. Andreas Savakis, Professor and Department Head
Primary Advisor – R.I.T. Dept. of Computer Engineering

_____________________________________________                ______________
Dr. Marcin Lukowiak, Assistant Professor
Secondary Advisor – R.I.T. Dept. of Computer Engineering

_____________________________________________
Dr. Andres Kwasinski, Assistant Professor
Secondary Advisor – R.I.T. Dept. of Computer Engineering
Thesis Release Permission Form

Rochester Institute of Technology
Kate Gleason College of Engineering

Title: Telemedicine Patient Identification with RFID;
An Embedded Approach

I, ONDER AKSOY, HEREBY GRANT PERMISSION TO THE WALLACE MEMORIAL
LIBRARY TO REPRODUCE MY THESIS IN WHOLE OR PART.

_________________________________
Onder Aksoy

_________________________________
Date
Acknowledgements

Many people have contributed and provided me with valuable feedback in doing this work. I would like to thank my advisor, Dr. Andreas Savakis, and committee members, Dr. Marcin Lukowiak and Dr. Andres Kwasinski.

Also, I would like to thank the following people:

- Greg Ratzel for his contribution of development tools and help in clarifying technical material.
- Steve Ortiz for his help with Borland.
- Richard Tolleson for his help with the logistics of my work.
- Pam Steinkirchner for her help with the logistics of my work.

Finally, I would like to thank my family, Kathleen, Bonnie, Joshua, and Jaykob for their unconditional support and encouragement throughout the entire process. They gave me the drive needed to complete my work.
Abstract

Radio Frequency Identification (RFID) has potential for application in the new field of telemedicine, as the use of radio waves offers advantages over traditional optical technology such as bar codes. Radio waves are not limited by line of sight, they can penetrate objects and communicate in a wireless fashion. However, the same advantage is also the inherent weakness, as radio waves are susceptible to attack. Ongoing efforts have identified forward secure chain hashing as a viable security protocol for RFID authentication. Today’s typical RFID communications take place with the “host-reader-tag” arrangement where the computational requirements are performed by a back end server system which holds all the intelligence and houses all records for an entire facility. One server can easily utilize multiple readers, but a compromise of this single system could have serious ramifications. Why not make a smaller system that is more robust and tolerant of intrusion. This can be achieved by implementing a stand alone reader that relies only on itself. We propose a server-less system that can accomplish the same results. Because our enhanced reader does not require a server to perform its function, if any readers are breached it only impacts that specific reader, not the entire server. By eliminating the resource heavy server device, we can yield a more robust overall system. We have selected a forward secure protocol to implement on an embedded platform that will be able to authenticate a tag without the resources of a back end server.
TABLE OF CONTENTS

THESIS RELEASE PERMISSION FORM ........................................................................................................ II
ACKNOWLEDGEMENTS ............................................................................................................................ III
ABSTRACT ................................................................................................................................................ IV
TABLE OF CONTENTS .......................................................................................................................... V
LIST OF FIGURES .................................................................................................................................. VIII
LIST OF TABLES ....................................................................................................................................... IX
LIST OF EQUATIONS ............................................................................................................................ IX
GLOSSARY ................................................................................................................................................. X

CHAPTER 1  INTRODUCTION .................................................................................................................. 1
  1.1. DESCRIPTION .............................................................................................................................. 1
  1.2. RADIO FREQUENCY IDENTIFICATION .................................................................................... 2
  1.3. TAGS ........................................................................................................................................... 3
      1.3.1 Active tags .......................................................................................................................... 4
      1.3.2 Passive tags ....................................................................................................................... 5
  1.4. READERS .................................................................................................................................... 5
  1.5. TELEMEDICINE .......................................................................................................................... 6
  1.6. CHALLENGES ............................................................................................................................. 7
  1.7. OVERVIEW .................................................................................................................................. 8

CHAPTER 2  SYSTEM CONSIDERATIONS ............................................................................................ 9
  2.1. TYPICAL RFID SYSTEM ............................................................................................................ 9
  2.2. RFID SYSTEM CONFIGURATION OPTIONS .......................................................................... 10
      2.2.1 Cell phones ....................................................................................................................... 11
      2.2.2 Internet ........................................................................................................................... 11
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.3.2 Reset</td>
<td>42</td>
</tr>
<tr>
<td>6.3.3 JTAG</td>
<td>42</td>
</tr>
<tr>
<td>6.3.4 Fuse settings</td>
<td>42</td>
</tr>
<tr>
<td>6.3.5 Communication ports</td>
<td>43</td>
</tr>
<tr>
<td>6.4. DATA STRUCTURES</td>
<td>44</td>
</tr>
<tr>
<td>6.4.1 Configuration registers</td>
<td>45</td>
</tr>
<tr>
<td>6.5. MESSAGING FORMATS</td>
<td>46</td>
</tr>
<tr>
<td>6.5.1 CRC</td>
<td>46</td>
</tr>
<tr>
<td>6.5.2 Tag data programming</td>
<td>47</td>
</tr>
<tr>
<td>6.5.3 Get UID</td>
<td>48</td>
</tr>
<tr>
<td>CHAPTER 7 TESTING AND RESULTS</td>
<td>50</td>
</tr>
<tr>
<td>7.1. ENCRYPTION RESULTS</td>
<td>50</td>
</tr>
<tr>
<td>7.2. SCAN RANGE</td>
<td>52</td>
</tr>
<tr>
<td>7.3. SYSTEM PERFORMANCE</td>
<td>53</td>
</tr>
<tr>
<td>7.3.1 Computation time</td>
<td>53</td>
</tr>
<tr>
<td>7.4. SYSTEM COST</td>
<td>56</td>
</tr>
<tr>
<td>CHAPTER 8 CONCLUSION</td>
<td>59</td>
</tr>
<tr>
<td>8.1. CLOSING REMARKS</td>
<td>59</td>
</tr>
<tr>
<td>8.2. AREAS FOR FUTURE RESEARCH</td>
<td>60</td>
</tr>
<tr>
<td>BIBLIOGRAPHY</td>
<td>61</td>
</tr>
</tbody>
</table>
List of Figures

FIGURE 1.1. ADHESIVE RFID LABEL USED IN PACKAGING INDUSTRY (1.5 X 1.7 INCHES) ........................................ 4
FIGURE 2.1 BASIC RFID SYSTEM IN USE TODAY .................................................................................................. 9
FIGURE 2.2 TRANSMISSION PATH OPTIONS ....................................................................................................... 10
FIGURE 3.1 SELECTED TRANSMISSION CONFIGURATION .................................................................................. 13
FIGURE 3.3 COMPLETE RFID READER DEVICE ................................................................................................. 14
FIGURE 3.4. ATMega64 MOUNTED ON TQFP TO DIP ADAPTOR ........................................................................ 16
FIGURE 3.5. LED BANK FOR VISUAL AID ......................................................................................................... 17
FIGURE 3.6. RW-210 MODULE ....................................................................................................................... 18
FIGURE 3.7. TAG MEMORY STRUCTURE ........................................................................................................ 19
FIGURE 3.8 HN-550 WIRELESS RADIO ........................................................................................................ 20
FIGURE 4.1 HASH FUNCTION ............................................................................................................................ 23
FIGURE 4.2 RFID PRIVACY SCHEME ................................................................................................................ 24
FIGURE 4.3 SHA-1 GENERATION ..................................................................................................................... 25
FIGURE 4.4 MD5 BLOCK DIAGRAM .................................................................................................................. 26
FIGURE 5.1 SCREENSHOT OF FSHASH CONSOLE APPLICATION ................................................................... 30
FIGURE 5.2. SHA-1 DIGEST FOR MESSAGE “Test” ............................................................................................. 31
FIGURE 5.3. MD5 DIGEST FOR MESSAGE “Test” ................................................................................................. 32
FIGURE 5.4. FORMATTED MD5 RESULTS FOR DIGEST “Test” ........................................................................... 32
FIGURE 5.5. AVR® JTAGICE mkII ..................................................................................................................... 33
FIGURE 5.6. SERIALCOMM GUI ..................................................................................................................... 34
FIGURE 5.7. SERIALCOMM PROGRAMMING OPTIONS .................................................................................... 35
FIGURE 5.8. NEW TAG ENTRY .......................................................................................................................... 36
FIGURE 5.9 AUTHENTICATION FAIL .............................................................................................................. 37
FIGURE 5.10 AUTHENTICATION CONFIRMED ................................................................................................. 37
FIGURE 6.1 READER STATE DIAGRAM ............................................................................................................. 39
FIGURE 6.2 HIGH LEVEL PROGRAM FLOW CHART ........................................................................................ 41
List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Tag memory assignments</td>
<td>28</td>
</tr>
<tr>
<td>6.1</td>
<td>ATmega64 data communications setup</td>
<td>44</td>
</tr>
<tr>
<td>6.2</td>
<td>NV memory configuration register</td>
<td>45</td>
</tr>
<tr>
<td>7.1</td>
<td>Prototype cost</td>
<td>57</td>
</tr>
</tbody>
</table>

List of Equations

\[ h(h(h(x))) \equiv h^3(x) \]  \hspace{1cm} \text{Equation 1} \hspace{1cm} 23
<table>
<thead>
<tr>
<th><strong>Glossary</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CRC</strong></td>
<td>Cyclic Redundancy Check</td>
</tr>
<tr>
<td><strong>DIP</strong></td>
<td>Dual in-line package</td>
</tr>
<tr>
<td><strong>GUI</strong></td>
<td>Graphical user interface</td>
</tr>
<tr>
<td><strong>IDE</strong></td>
<td>Integrated development environment</td>
</tr>
<tr>
<td><strong>I/O</strong></td>
<td>Input/Output</td>
</tr>
<tr>
<td><strong>ISO</strong></td>
<td>International Organization for Standardization</td>
</tr>
<tr>
<td><strong>JTAG</strong></td>
<td>Standard written by Joint Test Action Group, associated with IEEE 1149</td>
</tr>
<tr>
<td><strong>NV</strong></td>
<td>Non-volatile memory, retains contents after powering cycle</td>
</tr>
<tr>
<td><strong>PCB</strong></td>
<td>Printed circuit board</td>
</tr>
<tr>
<td><strong>PRNG</strong></td>
<td>Pseudo random number generator</td>
</tr>
<tr>
<td><strong>SMT</strong></td>
<td>Surface mount technology</td>
</tr>
<tr>
<td><strong>UID</strong></td>
<td>Unique Identification</td>
</tr>
<tr>
<td><strong>USART</strong></td>
<td>Universal Synchronous-Asynchronous Receiver/Transmitter</td>
</tr>
</tbody>
</table>
Chapter 1 Introduction

1.1. Description

Many people have experienced the commercial use of encrypted RFID in today’s commerce, such as the toll collection system “EZPass” or the “SpeedPass” payment system by Exxon-Mobil. SpeedPass uses the Digital Signature Transponder (DST) manufactured by Texas Instruments (TI). DST is a cryptographically enabled passive RFID device using a block cipher to implement a challenge-response authentication scheme. Despite using a cipher text to exchange data with the reader, the fact remains that the data interaction was performed on an open broadcast rendering the unprotected tag vulnerable to any compatible reader. This authentication protocol was defeated in 2005 [5] with a TI evaluation kit.

The operational premise of RFID is based on wireless transmissions over open radio waves, which is also the appeal for unauthorized access by an adversary. Acknowledging the possibility of attack, we shall consider the wireless exchange as compromised and include it in the development parameters. These parameters also consider tag anonymity, the ability to prevent tag tracking, low cost of computational resources, and means to keep previous information secure going forward. We explore previous works that have successfully addressed these criteria. The work in [6] proposes a matrix algorithm which is adaptable to the embedded platform, but passive tags lack the ability to operate a timer as specified, while [3] proposes an XOR scheme sharing random keys in a common list which require frequent overwriting of the complete list has failed to maintain security when not updated by the users.
In [1], [21], Ohkuno et al. state the criterion for a secure system includes indistinguishability, non-tracking, low-cost and forward-secure protocol. They propose the use of a one way function to circumvent the adversary tampering and implement a protocol that ensures the users privacy using a hash scheme. This method satisfies the low cost requirement and is adaptable to RFID use.

Luo et al. build upon [1] by adding on mutual authentication protocol [2]. This work describes how to make the tag output non-constant using a randomized key thereby securing the data interaction between the reader and tag. Though the work is sound, [2] needs to synchronize the reader and tag using a PRNG, which is cost prohibitive for implementation on a low cost tag, therefore, was not considered for our implementation.

Of the work researched, [1] has the best approach for our proposed thesis. We will implement an embedded system using an 8bit microcontroller as an alternative to the traditional back end server that can be as effective in the RFID information exchange while limiting the potential of wide scale data breach. The work in this thesis will contribute to the security concerns stated by providing a lower cost alternative to the back end server setup that is able to keep information previously sent from being revealed in the future.

1.2. **Radio Frequency Identification**

RFID has been in existence since WWII but was too costly to be considered for widespread use in industry. In recent years the acronym has become more commonplace and come forward to be an acceptable means of identification technology. As important technology eventually does, RFID has found its way in to commerce and is being
integrated into many areas in various forms. One may be using RFID in daily activities and not be aware of it. Libraries are using the technology for checkouts, shipping centers are using the technology for package tracking or work environments use RFID in the employee identification badges that grant building access to its personnel or perhaps a major credit card is embedded with a RFID device that does not need to be swiped to make a transaction.

In basic terms, RFID is a means to identify of one or more objects with the use of radio frequency waves. To make use of RFID technology there are three essential components required, these systems are discussed further in Section 2.1.

1.3. Tags

Tags, sometimes called transponders, are simple integrated circuits combined with an antenna that is tuned to receive a specific radio frequency and has the ability to be programmed with data for identification. A tag will receive a carrier signal from the reader and reflect a wave back to the reader encoded with the data requested. Tags respond to signals from any device that provides the correct modulation and instruction.

Tag memory can be read-write, read-only or “write once, read many” (WORM). Read-write tags have a serial number encoded that may not be modified and have additional blocks of data to store information available to the end user. Read-only tags have data programmed on them during the manufacturing process and can never be modified. WORM tags can have a non-modifiable serial number written to them once that is read numerous times.
Tags are available in a wide variety of form factors that range in size from the size of a grain of rice to a standard sheet of paper. The most common frequencies are low-frequency at around 125 KHz, high-frequency at 13.56 MHz and ultra-high-frequency at 860-960 MHz. Each frequency range has its advantages and is better suited for the particular applications where they will be deployed.

![Adhesive RFID label used in packaging industry (1.5 x 1.7 inches)](image)

All tags can be generalized under two categories, active and passive, described in the next sections.

1.3.1 Active tags

Active tags are powered with an internal battery to increase the effective operating range and usually support features that passive systems do not, such as analog sensors. Although the active tag is capable of greater range and additional functionality, there are several disadvantages. One limitation is the larger physical size requirements, and more importantly, its finite service life dictated by the battery and associated maintenance costs.
1.3.2 Passive tags

Passive tags are more prevalent and use a coiled antenna to create a magnetic field by capturing the readers radiated RF signal. The field is used to power up the tag circuitry and respond to the reader. Since passive tags rely on the reader for their operating power, they can potentially be held in service indefinitely. Passive tags are generally small in size and have a lower cost compared to active tags.

1.4. Readers

Readers, also known as interrogators, are transceiver devices that are used to interface with the tags through the antenna. Chip manufacturers have started to offer IC’s that consolidate much of the analog circuitry making readers compact, easy to incorporate and affordable. Readers are more of a relay device than a computing device; they require an intelligent host to instruct them on their actions and to make use of the data received from a tag. To retrieve the unique data on a tag, the tag must be scanned and decoded by a reader.

Scanning is a process initiated by the reader emitting a radio frequency (RF) signal that travels a short distance. When the tag passes through the electromagnetic field of the antenna, it is activated and performs the function requested by the reader. The radio waves backscattered from the tag are then converted to digital information by the reader circuitry and processed accordingly. We will use a transponder, which is a passive tag capable of read/write memory operations in the high frequency range.
1.5. **Telemedicine**

Telemedicine is the use of communication technology for the delivery of medical consultation or procedures at a distance. The appeal of telemedicine is that it has the ability to bring primary and specialty medical care into remote areas. The potential to help patients who live hours from basic medical care services, and would otherwise not be able to get to a medical facility, is enormous; it enables the patient to directly access quality medical professionals without leaving their communities. Telemedicine is not only restricted to patient use, medical specialists in off-site locations would be able to provide consulting services to onsite staff.

One example of the applications for telemedicine is an assisted living facility. The assisted living facility would benefit from promoting and offering virtual office visits with medical professionals from the facility. This is good for the living facilities ability to attract residents and increases the potential number of care sessions the resident could receive without actually leaving the facility. There is no need for the patient to be transported to the care giver and eliminates the logistics involved for the transportation. The virtual visit can also reduce the stress for the patient. Perhaps the patient has a condition that they wish to keep private as not to burden family members or they simply want to keep their affairs to themselves. Or perhaps the care giver at the living facility is a general practitioner or physicians assistant and needs to consult with a specialist; this ability would greatly increases the care available to the residents.

Another example is telepharmacy services. Telepharmacy can offer remote pharmacist controlled dispensing systems. This ability can bring pharmacist counseling
to the point of care and provide real-time medication dispensing. Have you ever had a prescription filled and not used the entire quantity? This could reduce the wasted product and save money for both patient and provider. Telepharmacy could expand the service area without the expense of traditional “brick and mortar” buildings and maximize staff utilization. One pharmacist can serve many locations rather than one per building which again saves money and resources.

Telemedicine enables the treatment process to begin much faster than traditional care requiring the patient to travel sometimes long distances to be physically present at a care facility.

1.6. Challenges

Although technology forecasts better health care, advances in the field are not without concerns. One major obstacle has been the issue of safeguarding the integrity, security, and privacy of healthcare information that is transmitted and stored electronically when using telemedicine services. There is a need for a reliable means of ensuring patient confidentiality in the telemedicine data exchange. This shortcoming is a primary concern when used in the context of telemedicine. A security gap exists in identity verification when the patient is not physically present to confirm that the patient is indeed who they claim to be.

Currently the method of disclosure in healthcare holds the patient identification confidential, not the patient care information. Keeping with the practiced disclosure method of the day, the focus here is on the security of the patient’s identity for which the tag is issued, not the entire system from tag to server. This thesis investigates the reader-
to-tag information exchange and implements a light weight security protocol to safeguard the patient’s identity.

1.7. Overview

The aim of this thesis is to explore the use of an established hash function algorithm suitable for service in a traditional server-reader-tag system and apply that concept to an embedded system without loss of effectiveness. Chapter 2, System Considerations, discusses the various configurations possible for our proposed RFID system. Chapter 3, System Design, discusses our selected system configuration. Chapter 4, Security Protocol, provides overview of what forward secure chain hashing is and reviews the work done by [1][21] which is the basis of this thesis. It also discusses the selected hashing algorithms and how our RFID tags are authenticated. Chapter 5, Work Environments, reviews the console application fshash, the embedded firmware, the user interface SerialComm, and the respective development tools used. Chapter 6, Embedded Code, presents a high level software state diagram for the firmware, the firmware, data structures, and messaging formats. Chapter 7, Testing and Results, provides summary of results for the range of operation, system performance results and cost of the system. Chapter 8, Conclusion, summarizes the accomplishments of this work and provides suggestions for future enhancements.
Chapter 2  System Considerations

2.1.  Typical RFID system

RFID systems deployed today have at minimum three components consisting of a host, reader and tag. The host or back end server provides the intelligence of the system and stores a data base of values that are associated with the tags for identification purposes. The reader sends requests and demodulates responses and the tag is attached to the object to be identified. There can be one or many readers configured for use with the single host and there are many tags. The amount of information stored in the database is only limited by the resources of the back end server.

![Figure 2.1 Basic RFID system in use today](image)

The typical system is a good choice for non-critical applications such as the packaging industry where it is not vital to conceal a customer’s shipping information, but this arrangement is not well suited for our use in telemedicine where patient ID must be hidden. In the next section, we will investigate a more appropriate setup for use with our RFID system.
2.2. **RFID system configuration options**

We plan to implement a system that does not need the back end server a typical system requires. To do this, we will use a microcontroller to provide the intelligence and communications with the peripheral components. Most microcontrollers have a serial communication port which will allow for interfacing with common devices.

The reader, tag and display make our system. The main component of our system, the reader, consists of the microcontroller, a RFID module and an integrated antenna. The tags are purchased read/write capable passive tags and there are several choices of displays to receive the reader response. These three are the most common, but this is not an exhaustive list. As shown in Figure 2.2, the display can be a computer that is connected to the internet, wireless transmitters, or cell phones. Essentially the transmission to be received from the reader needs to have a path back to the user.

![Diagram of transmission path options](image)

*Figure 2.2 Transmission path options*
2.2.1 Cell phones

In order to use cell phones for display, there must be an adaptor from cell phone to reader and platform specific software on the receiving cell phone. The microcontroller can provide ASCII text to the reader side cell phone via a serial-to-USB device which can then be forwarded to the user’s cell phone. The advantage of this is that there are many cell phones currently in use and the physical location to call in or provide an office visit can be almost anywhere. The communication medium offers great flexibility, however, the disadvantage is that there must be cell service in the area. Of course there are satellite service options available, but our application was targeted towards remote areas which more often would not have such resources available.

2.2.2 Internet

To use the internet there must be a serial-to-Ethernet device on the reader to route the packets to the user’s/care providers internet enabled display. These devices are readily available as add on modules from many vendors and can be incorporated in to existing designs easily as they use the same serial output from the microcontroller without modifications. This capability would literally enable world wide deployment and may possibly appeal more to the care provider than the patient. Similar to the cellular service discussed previously, it implies that there must be an internet service provider and access to the internet, which is not typically associated with the remote areas targeted.
2.2.3 Wireless transmitter

To use the transmitter there must be a means to forward the readers ASCII output, ideally with a DB9 serial connection. This would be a simple device that could be attached or interchanged with a different radio or possible mixed with radios of different models. Radios do not need any specific connection services, as required cell phones or the internet, and they provide a good option for service in remote areas. However, the radio may be limited by the distance it can transmit or the geography of a region where it is deployed.

2.2.4 Transmission encryption

It is possible to encrypt this transmission by selecting a microcontroller with a built in encryption engine, such as the AVR® XMEGA series from Atmel®. These devices have cryptography support for AES and DES. This would add another level of security to the data exchanged and prevent attackers from obtaining the information. To use this functionality, an appropriate utility must be used on the receiver side to correctly decrypt the responses.
Chapter 3  System Design

From the several options and possible configurations we have shown, we have decided to employ a system that uses a wireless radio. The interface to the radio is simple a DB9 connection and the radios do not require the additional overhead of specific software for the internet or cell phone device. Also, the microcontroller we have selected does not have an encryption engine; this decision was made because of the frequency hopping operation of our chosen radio and inventory on hand.

3.1. RFID reader

The reader is considered a single component, but it is comprised of several components; a block diagram of the device is shown in Figure 3.2. The details for each are listed in the Sections 3.1.4 through 3.1.3.
3.1.1 ATmega64 microcontroller

The microcontroller is replacing the back-end server in the traditional RFID system. This micro is capable of handling the same computational workload needed, but
is scaled down in its available memory. We selected the Atmel® ATmega64 microprocessor [16] for our reader for the following attributes:

- 64k bytes of reprogrammable flash memory
- 4k bytes of SRAM
- 2k bytes of NV memory rated for 100,000 write/erase cycles
- SPI interface for in-system programming
- JTAG on-chip debug support
- two dedicated programmable serial communication ports
- software selectable clock frequency
- programming lock for security
- rated for 100 year data retention (normal ambient conditions)

In addition to the no-cost software development tool, the primary reason for selecting this component was the anticipated size of firmware to be used. It was expected that there would be need for a substantial amount of memory for the porting of the two hashing algorithms used and a large amount of memory for text strings used in the host communications. The chip comes in a 64pin TQFP package and readily available from major distributors.
Communications between the ATmega64 and RW-210 are not encrypted. This is considered internal communications for the reader device and not exposed to the outside world. These communications take place on circuit board traces and are not sent wirelessly, therefore they are considered secure.

3.1.2 LED visual indicators

There are four LED lights on the reader for the benefit of visual confirmation of the various operations performed. The LEDs are controlled by the microcontroller, as the code enters the different states.

LED0: Indicates PROGRAMMING MODE, this LED will light up and remain lit until the patient ID and tag programming is completed. This is the LED placed adjacent to the reset pushbutton.

LED1: Indicates when the reader is expecting a tag to be scanned, it will flash once per second until a tag is scanned.
LED2: Indicates whether the tag authentication was successful. It will remain lit for a one long second duration.

LED3: Indicates whether the tag authentication failed. It will remain lit for a one second long duration.

When the system is powered on, all four LEDs will flash in order (3 to 0, 0 to 3) several times to indicate the system has been successful initialized and is ready for use.

3.1.3 Serial Converter

This is the one of the standard components used in industry for serial-to-TTL conversions. This converter, a Maxim MAX233CPP, or any other converter, is required for interfacing the reader with the outside world and in our configuration, the SerialComm user interface running on our display. Given this communication is sent to the outside world, it can be intercepted at this physical point in the circuit. This is the portion that would benefit from the encryption engine that would be available on the microcontroller.
3.1.4 RFID module

RW-210 RFID Reader/Writer from APSX [15] provides the interface functionality for the ISO15693 tags we are using. The module requires TTL level inputs and has an integral antenna that operates at a frequency of 13.56MHz. The baud rate for this module is 19200bit per second and it has a read/write range that is a 4 inch radius (cone shape).

![Integrated antenna](image)

Figure 3.6. RW-210 module

The wireless data exchange between the RW-210 and a tag is not encrypted and therefore is susceptible to attack, however, the short range would make this very difficult as the patient would be able to visually detect a sniffing device placed close enough to be effective. Though the communication is wireless, we will assume that this is not a great concern.

3.2. RFID tag

The RI-TH1-CB1A tag from Texas Instruments [11] is an ISO 15963 compliant device and is approximately the size of a standard credit card. It has a maximum storage
capacity of 64 blocks of raw data, where each block holds 32-bits for a total capacity of 2048 bits. Each tag is assigned a 64-bit UID by the manufacturer at the time of production, which may not be modified. The tags are readily available and have a cost of a few dollars.

![Figure 3.7. Tag memory structure](Source: TI data sheet for RI-TH1-CB1A [11])

### 3.3. Wireless Radios

The wireless transmitters we selected are the HN-550. These are 2.4GHz wireless data modems [24] and are ideal for this particular application, as they have a 9-pin serial interface and are frequency hopping devices. This means that unless the hopping pattern is known, the transmission can not be intercepted as a whole transmission, only portions and thus offers a level of protection from an attack. Another desirable feature is that the HN-550 is capable of point-to-point or point-to-multipoint networks making it possible to
have multiple users in the same location without interference from one another or increasing the transmission range by hopping the transmissions through multiple units.

![HN-550 wireless radio](image)

**Figure 3.8** HN-550 wireless radio

We have described what the key components of the system are, in the next chapter we will focus on the specifics pertaining to security protocol we implemented.
Chapter 4    Security Protocol

4.1.  Protocol selection

Several security protocols were examined and considered for implementation as reviewed in Section 1.1. The driving factors for the selection of chain hashing are;

- Light weight protocol
- Protocol that can eventually be implemented on tag circuitry
- Computations will be contained in limited resource microcontroller

A light weight algorithm, meaning an algorithm having less complexity in order to reduce overhead, is needed to achieve the proposed goal. There are other approaches to achieve secure communications, such as PRNG and block matrices. However, to comply with the forward secure component, a PRNG would need to be synchronized between reader and tag which would require additional circuitry and power. Block matrices are not as strong as the forward secure cryptographic chaining hash.

Another factor is the tag itself. Although current technology has not done so at the present time, this protocol must be able to be realized in to the tag circuitry. The small size of the tag IC must be retained therefore the gate size must be kept as small as possible. In [25] it was shown by using a 0.13μm CMOS standard cell library that the SHA-1 circuit would be approximately 8k to 10k gates and the MD5 circuit would use approximately 10k to 18k gates. These were gate counts for independent circuits; it may be possible to combine common components to use fewer gates which may yield a smaller unit cost as well as have small power requirements to carry out the necessary
hashing operations. The gate layout is not in the scope of our thesis and is not investigated further.

The digest computation is to be entirely performed on the reader device which is an 8-bit microcontroller. Microcontrollers are not designed like a back end servers nor do they have commensurate resources, therefore the protocol must be straightforward yet provide security at a sufficient level. Another advantage, the cryptographic hashes generate result of known length, this is valuable when using an 8-bit device with limited memory storage capabilities.

Based on our needs and the extensive supporting research in [21], the decision to use chaining hashes was made.

4.2. **Forward-secure chain hashing**

Forward security is the confidentiality of previous messages, meaning that data transmitted today shall remain secure in the future, such that even if the future message contents are discovered, there is no way to correlate the current and future digests.

One potential method to achieving forward security is to use one of the fundamental primitives in cryptography known as a *hashing function*, sometimes called a “one-way” hashing function. The hashing function maps a bit string of arbitrary length, known as a *message*, to a bit string of fixed length, called a *digest*, thereby preventing unauthorized retrieval of the original bit string. The way this algorithm operates makes it improbable to derive the message from the digest.
To take the one-way hash a step further, the algorithm can be applied successively to a message, which is known as hash chaining. This method can easily produce multiple one-time keys from a single message that are difficult to decode. Equation 1 shows function $h(x)$ which is a hash chain of length 3.

$$h(h(h(x))) = h^3(x) \quad \text{Equation 1}$$

Reference [12] defines “A one-way hash function is a hash function that satisfies the property that $f(m)$ is “easy” to compute for all $m \in M$, but for randomly chosen $c$ in the image of $f$, finding an $m \in M$ such that $c = f(m)$ is computationally infeasible, namely we can easily compute $f$, but it is computationally infeasible to compute $f^{-1}$.”

In [1], Ohkubu, et al. propose using an initial message, $s_i$, apply hash chaining and update the message for the next transaction. Here the tag would store the information, $s_i$, send the digest $a_i = G(s_i)$, and then renew its secret to $s_{i+1} = H(s_i)$ based
on the previous message. Forward security is implemented by making use of two one-way hash functions, H and G, to calculate the digest.

![Diagram](image)

**Figure 4.2 RFID privacy scheme**

This scheme provides the forward security we require. If \( a_i \) is discovered, G is a one-way hashing function, therefore \( s_i \) can not be revealed. H is also a one-way hashing function, so if the tag contents are discovered, \( s_i \) can not be distinguished from \( s_{i+1} \).

On the server, a database of tag identifications and messages, \((ID, s_i)\), is maintained. When the server receives the tag output, it will calculate the \( G(H(s_i)) \) and would verify the received digest matches the calculated digest.

### 4.3. Protocol application

Our implementation does not require a back end server, and the computational requirements of the protocol are contained in the reader’s microcontroller. The reader is not intended to service a vast number of patients and because of that does not need to have a large memory section. The tag/patient database is to be stored in the
microprocessors non-volatile memory so it can be retained through power cycles. The one-way hash functions chosen were primarily with the digest length as criterion. We selected the SHA-1 and MD5 functions to demonstrate the operation of the our chosen algorithm, other functions, for example SHA-2 or the winner of the SHA-3 competition, could be used in place of the SHA-1 to provide additional strength. The length of the second function needed to be as short as possible while balancing encryption strength.

The first hash is done using the SHA-1 functions designed by the NSA [7],[8]. With a maximum message length of \(2^{64} - 1\), the digest produced is 160-bits long. Our input message is limited to 9 ASCII characters, but may be increased if warranted.

![SHA-1 generation Diagram](image)

**Figure 4.3** SHA-1 generation
The second hash is done using the MD5 function [9] designed by R.L. Rivest commonly used for file download integrity. MD5 produces a 128-bit digest in our application and uses the SHA-1 digest as the input message. The shorter digest length of 16 bytes was desired to maximize the number of digests to be programmed on a single tag. If a different function is implemented in this part, the effect would be that the total number of digests stored on the tag would be reduced as the tag memory is fixed.

![MD5 block diagram](image)

Figure 4.4 MD5 block diagram

The tags are not capable of computation, only data storage; therefore, we can not have a fully compatible system that is specified by [1] where the tag generates the digest in real time when requested. In lieu of an intelligent tag, we will program and store the MD5 digest on the tag to be scanned by the reader.

4.4. Tag authentication process

The process of authentication begins with the programming of the reader. At the time of commissioning, a tag is scanned to obtain its valid UID. After verifying the tag
has not been entered previously, the patient ID number, message and number of hashing iterations to chain are all entered in and stored to the reader’s microprocessor memory. By allowing any length message, up to 9 characters, and an unknown number of iterations to chain ranging from 16 to 133, the message will remain secure from discovery.

The tag is only able to store 16 digests at any time, and to generate the digest data, a separate console application called \textit{fshash} is used. The \textit{fshash} application requires the same message and a number of iterations to generate the final 16 digests in the series of hashes. These resulting 16 digests can be programmed on to the tag at any time using the reader and GUI interface.

Once in service, the reader is remotely instructed via a RS232 serial port to authenticate a tag and signals the patient to place the tag to be scanned. The identity of the tag is checked against the entries in memory, and if the UID is not found, the process ends. If the entry is present, the reader reads the stored values for the message and number of iterations and calculates $h(h^{n-1}(\text{message})) = h^n(\text{message})$. This is the digest we are expecting to read on the tag. The reader then reads from memory the tag block number that corresponds to the 16 digests on the tag, and issues a command to the read tag memory contents of that location.

For example, we have entered the following:

\begin{verbatim}
Message: test
Iterations: 38
\end{verbatim}

The application provides digests 22 through 38 and these are programmed on to tag memory mapped as $22 \rightarrow \text{block1}$, $23 \rightarrow \text{block2}$, successively to $38 \rightarrow \text{block16}$. On
request $h(h(h^{36}(\text{test}))) = \text{digest iteration 36}$, a request for the contents of tag memory block_14 is issued.

<table>
<thead>
<tr>
<th>Memory block</th>
<th>Digest</th>
</tr>
</thead>
<tbody>
<tr>
<td>block_1</td>
<td>$h^{22}(\text{test})$</td>
</tr>
<tr>
<td>block_2</td>
<td>$h^{23}(\text{test})$</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>block_16</td>
<td>$h^{38}(\text{test})$</td>
</tr>
</tbody>
</table>

Table 4.1 Tag memory assignments

The reader compares the calculated digest with the one received from the tag. If the information is a match, an LED validates the patient using the reader and a confirmation text is sent to the remote user. Following the positive confirmation, the iteration number and tag block numbers on the reader memory are decremented by one and stored for the next authentication request. The current digest (the digest just authenticated) is updated with next digest in line for subsequent authentications. For example, in Table 4.1 the last digest to be authenticated is $h^{22}(\text{digest})$. When a authentication for $h^{38}(\text{digest})$ is successfully completed, the digest is updated with $h^{21}(\text{digest})$, the next digest needed following the data in block_1. This complies with the forward secure requirement to overwrite the digest that was immediately used.

If they do not match, an LED indicates the failure to the patient and a failure text is sent to the remote user. In this case, the reader memory is unchanged. All of the information exchanged is reported back to an authorized user via the RS232 serial port.
Note that the serial exchange between reader and display is not encrypted for use in this thesis, but can be implemented if desired.

We have explained the security protocol and how the authentication takes place, in the next chapter we discuss the various software tools and interfaces to use the RFID system.
Chapter 5 Work Environment

5.1. fshash

The console application, *fshash*, was written to assist in the tag digest programming. The application prompts for a message that can be any value up to 9 characters in length. The reason for this is that initially the patient’s Social Security Number was intended for use, however, any value may be assigned. The message entry whether intended as numerical or text, is used as an ASCII value, so it is possible to have control characters, such as LF (line feed), as a resulting value. To prevent this from occurring, we have, for the purpose of this thesis, limited the range to 16 through 133. ASCII value ‘z’ may have 133 increments before it will enter the control character values.

![Screenshot of fshash console application](image)

Figure 5.1 Screenshot of *fshash* console application
The SHA-1 and MD5 algorithms are publicly available and provided free of charge on several web sites. `fshash` will generate the digests we are interested in and write three output files, one for the SHA-1 digests, two for the MD5 digests.

The SHA-1 file, named “sha1-out.txt”, was used to verify the hash results and each digest is the input message for the following MD5 algorithm.

![Figure 5.2. SHA-1 digest for message “Test”](image)

The first MD5 file, “md5-out.txt”, is the entire digest as will be generated by the reader. This file is intended as useful information for the user.
Figure 5.3. MD5 digest for message “Test”

The second, md5-out-fmt.txt”, is a duplicate of the first with a delineating space written between each byte of digest for programming purposes. The data must be entered with this format to the interface application to program the tag. For convenience, it may now be copy/paste as is in to the SerialComm GUI user interface.

Figure 5.4. Formatted MD5 results for digest “Test"
fshash is written in the C language using the Code::Blocks ® open source programming environment [19].

5.2. Embedded firmware

The firmware for the microprocessor was written using AVRStudio4®, Atmel’s® free Windows IDE using the GCC compiler [17]. All source code development and debug was done using the AVR® JTAGICE mkII [18] from Atmel®, a development tool for On-chip Debugging with IEEE 1149.1 compliant JTAG interface. Chapter 5 details the primary components of the source code.

![AVR® JTAGICE mkII](image)

**Figure 5.5. AVR® JTAGICE mkII**

5.3. SerialComm

The user interface for the system, SerialComm, is a basic serial communications GUI application for use with the Windows operating system. This application is not specifically required as the instructions to the reader are simple text and hexadecimal inputs. The application was written using the C++ Builder 6 IDE from Borland®.
The lower left portion of the GUI is the host communication port selection options. When the application is launched, it searches for available all communications ports and presents only the active ports. The communications options, 38400bps, 8-N-1, have been selected as shown in Figure 5.6. The settings may be saved for future use.

The text box located above the Transmit button is where all user selections are entered. To send data, the user must click the Transmit button. When programming the tag digest data, the Tag data check box must be selected, and this will send the data as hexadecimal values.
5.4. Programming menu

The program option presents several selections. The user may choose to enter new tags, edit or remove existing tags, or program tag digests individually, or all blocks at once.

![SerialComm programming options](image)

Figure 5.7. SerialComm programming options

To enter a new tag the user must enter the following:

a. Tag number which in our version can be between 1 and 10.

b. The 9 digit patient ID

c. The secret key (also called message)

d. Number of iterations which in our version can be between 16 and 133

All entries are confirmed after each entry is made.
Once a tag is entered, it may be entirely removed or fields Patient ID, Secret Key or Iterations may be updated. Each update is confirmed as it is made. The tag digest is easily programmed. If the entire 16 blocks are selected, the user enters the data, checks “Tag data” and clicks the Transmit button. If single digests are selected, the user indicates which block to program, specifies the single digest to program, checks “Tag data” and clicks the Transmit button.

5.5. Authentication

To authenticate a patient, the user selects the authenticate option, the reader will indicate to the patient by flashing the LED and printing an asterisk every second on the SerialComm window. Once the patient places the tag on the reader, the authentication
initiates and indicates the results. The reader will indicate one of three possibilities, the tag was not found in memory meaning it was never entered, authentication failed, or authentication confirmed.

![Figure 5.9 Authentication FAIL](image)

![Figure 5.10 Authentication CONFIRMED](image)
Having articulated the software environments and their use, the next chapter will detail the RFID reader and tag at a system level.
Chapter 6  Embedded Code

The embedded source code is loaded into the microprocessor; this code dictates the behavior of the reader device. The following sections illustrate the essential components of the embedded code.

6.1.  State diagram

The RFID reader has the following normal operating states, READY, PROGRAM, and AUTHENTICATE. The RESET can be initiated at any time and will go the READY state. Transitions between states are governed by the following state diagram.

![Reader state diagram](image)

Figure 6.1  Reader state diagram
6.2. Program flow

The RFID reader system software is a repetitive cycle, the program flow is described in the chart in Figure 6.2. The system waits at the ready state until the user enters a request for action. The authentication action follows through with tag verification and returns to the ready state. The program action prompts for the user to select what to program and then processes that request. After programming, the system returns to the ready state.

6.3. Firmware

The RFID reader system is governed by a small event driven real time operating system. The system has soft time requirements and is not life critical. The notable attributes are discussed in the following sections.

6.3.1 Clock source

The reader circuit uses an external ceramic chip resonator at a frequency of 7.3728MHz. This component was selected because the frequency (4\times1.8432 \text{ MHz}) allows integer division to common to baud rates reducing the bit error rate in serial communications. The resonator has built-in capacitors and needs only to be connected to the XTAL1 (pin 24) and XTAL2 (pin 23) pins of the ATmega64 microcontroller. The fuse settings must be set accordingly. Section 6.3.4 provides additional details.
Figure 6.2 High level program flow chart
6.3.2 Reset

The reader may be issued a hard reset by pressing the only pushbutton in the circuit tied to the RESET pin. This action is an “absolute jump” command, the equivalent of cycling power on the reader. This will cause the system to begin execution at instruction 0x0000 and reset all I/O resources on the ATmega64. A reset command does not erase the contents of the NV memory.

6.3.3 JTAG

The ATmega64 is capable of JTAG interface, which allows for flash memory programming and in-circuit debug. This functionality is very important as the ATmega64 is a SMT part meaning it is not easily removable. DIP components may be removed by simply unplugging; this is not the case for SMT components. Once SMT components are soldered in place there are a finite number of times the component can be soldered and removed before the board pads are damaged. JTAG enables the embedded software development to continue while the microcontroller is still installed in the circuit. The modified code can be uploaded and tested as frequently as needed without the removal of the microcontroller. Four pins of the ATmega64 in Port F (PF5 – PF7) are required to attach the AVR® JTAGICE mkII to utilize the interface.

6.3.4 Fuse settings

The fuse settings are the configuration bits for the ATmega64 that allows the for the set up of the various aspects of the microcontroller such as boot flash size, enable
JTAG interface, the clock to run (external crystal, internal oscillator) and to save the EEPROM memory contents on power cycles. To permanently lock the microcontroller, the lock bits may be set, thereby preventing further access.

![Figure 6.3 ATmega64 fuse settings](image)

6.3.5 Communication ports

Two true dedicated serial communications ports are available in the ATmega64 microcontroller. One named USART0 is used for the host-reader communication, the other named USART1 is for the module-reader communications. Setup for each is shown in Table 6.1.
<table>
<thead>
<tr>
<th>PORT</th>
<th>LINE</th>
<th>DIRECTION</th>
<th>INTERFACE</th>
<th>USART</th>
<th>BAUD</th>
</tr>
</thead>
<tbody>
<tr>
<td>PE0</td>
<td>Rx</td>
<td>From</td>
<td>Host</td>
<td>0</td>
<td>38400</td>
</tr>
<tr>
<td>PE1</td>
<td>Tx</td>
<td>To</td>
<td>Host</td>
<td>0</td>
<td>38400</td>
</tr>
<tr>
<td>PD2</td>
<td>Rx</td>
<td>From</td>
<td>RW-210</td>
<td>1</td>
<td>19200</td>
</tr>
<tr>
<td>PD3</td>
<td>Tx</td>
<td>To</td>
<td>RW-210</td>
<td>1</td>
<td>19200</td>
</tr>
</tbody>
</table>

Table 6.1 ATmega64 data communications setup

The hardware interrupt service routine vectors, USART0_Rx_vect, USART1_Rx_vect, USART0_Tx_vect and USART1_Tx_vect, are defined in “avr/iom64.h” to indicate the reader has just received a character or finished sending the contents of the output buffer respectively. To transmit a message, we monitor the Tx complete interrupt, which is set when the last byte is sent out. On the receive side, the Rx complete interrupt is set when a single byte is received. To ensure that all serial data has adequate time to buffer, 20ms software delays are used where needed. This allows sufficient time for all packet bytes to be read in completely before any parsing is done.

6.4. Data structures

The tag data relates to the patient and this is stored in a struct in the microcontroller memory. The struct is shown in Figure 6.4. The information contains everything required for the authentication process.

```c
struct tag {
    BYTE uid[UID_LENGTH];
    BYTE patient_id[P_ID_LENGTH];
    BYTE key[KEY_LENGTH];
    BYTE n_val;
    BYTE next_dig_read;
    BYTE next_dig_prog;
};
```

Figure 6.4 Tag-Patient data structure

44
The first three struct members are set when the tag is commissioned and last three struct members are updated on each tag authentication. This information will remain intact between system power cycles. The struct is mirrored in the configuration registers.

6.4.1 Configuration registers

The RFID reader will have the following parameters stored in NV memory.

<table>
<thead>
<tr>
<th>Register Number</th>
<th>Register name</th>
<th>Size</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>uid</td>
<td>12</td>
<td>Unique ID</td>
</tr>
<tr>
<td>0x0C</td>
<td>patient_id</td>
<td>9</td>
<td>Patient’s ID</td>
</tr>
<tr>
<td>0x15</td>
<td>key</td>
<td>9</td>
<td>Input message</td>
</tr>
<tr>
<td>0x1E</td>
<td>n_val</td>
<td>1</td>
<td>Iteration number</td>
</tr>
<tr>
<td>0x1F</td>
<td>next_dig_read</td>
<td>1</td>
<td>Next digest to be read</td>
</tr>
<tr>
<td>0x20</td>
<td>next_dig_prog</td>
<td>1</td>
<td>Next digest to be programmed</td>
</tr>
</tbody>
</table>

Table 6.2 NV memory configuration register

uid unique identification number of the tag
patient_id patient’s ID assigned by the care facility
key input message to be encrypted
n_val iteration number for encryption
next_dig_read next digest to be read in authentication
next_dig_read next available tag block for programming

Figure 6.5 Configuration register definitions

There are 10 identical tag records allotted in our firmware, therefore each tag entry record will be offset by 32 bytes. The ATmega64 can hold 2k bytes of data in NV memory, giving a possibility of 60 tag records per reader if required. The total number of
6.5. Messaging formats

The following protocol format has been defined for the RFID reader to use in communications with the RW-210 module. All messages between the reader and RW-210 module must contain CRC bytes appended to the message.

6.5.1 CRC

The CRC-16 formatting is a basic XOR scheme using a “Preset Value” and a “Polynomial”. A number of bytes specified in the function parameters are processed in the CRC calculation and, once determined, the 16-bit CRC is returned. The pseudo code is shown in Figure 6.6.

Initialize the 16-bit remainder (CRC) to 0xFFFF

Loop:
XOR the 8-bit data byte with the CRC
Shift the CRC 1 bit to the right
Increment shift count

Is the bit shifted out to the right a 1 or a 0?
If 1, XOR CRC with generating polynomial
If 0, shift CRC right 1 bit

Repeat LOOP 7 more times

1’s complement CRC
Format CRC, swap high and low order bytes
Return CRC value

*Figure 6.6 CRC pseudo code*
This CRC is appended to any data communication going to and coming from the RW-210 module.

### 6.5.2 Tag data programming

To program the tag, an instruction must be sent with the format in Figure 6.7. The packet can hold 4-bytes of data. A digest is 16-bytes in length, therefore there are 4 separate packets required when programming a single digest. The full digest is parsed into four parts, appended with the CRC, and sent to the RW-210 module for transmission to the tag. The same function is used to program a single digest or the full 16 digests.

![Figure 6.7 Write tag block message format](image)

The individual bytes are defined as:

- **SOP**: Start Of Packet - indicates number of bytes in complete packet following this byte
- **Bytes Exp**: Number of bytes expected in reply
- **Flag**: ISO15693 specific, sets option flag, data rate and sub carrier
- **Block**: Tag block to write
- **Cmd**: ISO15693 tag instruction command
- **D1 – D4**: Data bytes 1 through 4
- **CRC1**: High byte of CRC
- **CRC2**: Low byte of CRC

![Figure 6.8 Tag message definitions](image)
To generate the correct CRC bytes, the first two bytes of the write command are excluded in the calculation. The format to forward to the CRC function is shown in Figure 6.9.

![Figure 6.9 CRC-16 format for tag block programming](image)

After determining the CRC, the result must be appended as well as bytes 0 and 1, prior to issuing the instruction.

### 6.5.3 Get UID

The UID may be requested from any tag by using one of two commands, the `get_uid` or `fast_uid`. The `get` command is formatted as shown in Figure 6.10

![Figure 6.10 Get UID instruction](image)
The get UID frame bytes are:

| SOP    | Start Of Packet - number of bytes following this byte |
| Bytes Exp | Number of response bytes |
| Flag    | ISO15693 specific, sets option flag, data rate and sub carrier |
| Cmd     | ISO15693 inventory command |
| Blk Num | Block number to read |
| CRC1    | High byte of CRC 16 |
| CRC2    | Low byte of CRC16 |

Figure 6.11 get_UID message definitions

The 12 byte response is the tag ID with the two CRC bytes appended at the end. The fast_uid command is 0xFA and it serves the same purpose as the get_uid command. The advantages of this command are convenience and speed, and there are 6 less bytes to send for each transmission. The response to fast_uid command is the same as the get_uid command.

Having described the user interfaces, the system components, and firmware operations in the previous chapters, we will deploy the system in our test environment and review the results in the next chapter.
Chapter 7  Testing and Results

7.1. Encryption results

Two hashing algorithms, SHA-1 and MD5, are used to produce the digest encoded on each tag. The firmware digest produced was compared against a digest produced using a freeware application called HashCalc [23].

Using the input message “TEST” with 32 iterations of hashing, the digests are shown in Figure 7.1. Applying the iterations will update the message to the ASCII string “TESt”. The last line in “sha1-out.txt” matches the value in HashCalc.
Figure 7.2 MD5 digest comparison
The file “md5-out.txt” shows the final digest for the input message (the digest from \textit{SHA-1}) “120b9b60d9eb0a0ef37cb334c9c37d213b34e24d”. The values generated by \textit{HashCalc} and the firmware are shown in Figure 7.2. The MD5 result is the final digest and is encoded on the tag at block 16.

7.2. \textbf{Scan range}

Scanning range of a high frequency RFID tag is less than 3 meters per the standard, however, this ideal and very dependant on the reader circuit parameters and environment in which the system is deployed.

<table>
<thead>
<tr>
<th>Name</th>
<th>Frequency Range</th>
<th>Wavelength Range</th>
<th>ISM Frequencies</th>
<th>Read Range for Passive Tags</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low frequency (LF)</td>
<td>30 – 300 KHz</td>
<td>10km – 1km</td>
<td>125 – 135 KHz</td>
<td>&lt;50cm</td>
</tr>
<tr>
<td>High frequency (HF)</td>
<td>3 – 30 MHz</td>
<td>100m – 10m</td>
<td>6.78 MHz, 8.11 MHz, 13.56 MHz, 27.12 MHz</td>
<td>&lt;3m</td>
</tr>
<tr>
<td>Ultrahigh frequency (UHF)</td>
<td>300 MHz – 3 GHz</td>
<td>1m – 10cm</td>
<td>433 MHz, 869 MHz, 915 MHz</td>
<td>&lt;9m</td>
</tr>
<tr>
<td>Microwave frequency</td>
<td>1 – 300 GHz</td>
<td>30cm – 1mm</td>
<td>2.44 GHz, 5.80 GHz</td>
<td>&gt;10m</td>
</tr>
</tbody>
</table>

\textbf{Figure 7.3 RFID read range}

(Source: \textit{How to Cheat at Deploying and Securing RFID} [20])

The RW-210 module is capable of reading at a maximum range of 7 inches according to the manufacturer [15]. When tested in an environment that has no metal or magnetic
fields (e.g. computer speakers) in proximity of the tag or module antenna, we achieved a scan distance of 3.5 inches. For our use in our deployment, this is acceptable. The shorter distance is sufficient since our application is a patient that is deliberately passing their officially issued tag across the reader antenna. An ancillary advantage is that the tag may not be maliciously scanned at long distance by an attacker.

7.3. System performance

The system was tested in a normal residential setting with average appliances and services. There may be additional circumstances (i.e. medical equipment) that could skew the results found here.

One of the HN-550 radios which we named the base was attached to the host PC via a serial port and the second radio named the remote was attached to the reader. First the radios were indoors, positioned 50 feet apart in a residential environment and worked correctly with no perceivable time delays in transmission. Next the radios were tested with one radio outside and the other indoors at a range of 100 feet, the system again performed correctly with no perceivable time delays in transmission. The HN-550 data sheet specifies a range of 1000 feet indoors, and, based on our testing, this range should be feasible with greater ranges possible using multiple radios to hop the transmissions.

7.3.1 Computation time

Computation and response time are of importance in the use of the system, as it must be able to confirm or deny patient identity in an expedient manner. After a tag is
scanned, our reader can perform real-time dual hash calculations, query the tag block for the digest, authenticate and display the results within a time frame of two seconds.

The longest operation is a response to the get_digest command. By examining the data transfer rate of this exchange, we will obtain a benchmark time to compare against. The communication rate between the microcontroller and RW-210 module is set at 19200bps. The get_digest instruction issued by the reader shown in Figure 7.4 is 8bytes long (64bits), so the time required is as follows.

\[
\text{Instruction} = 8\text{bytes} \times 8\text{bits} = 64\text{bits}
\]
\[
19200\text{bps} = 19.2 \text{bits per millisecond}
\]
\[
64\text{bits} \div 19.2 = 3.33\text{ms} \text{ (ideal transfer rate)}
\]

We show that approximately 3ms is needed for the microcontroller to send the get_digest instruction to the RW-210 module. Once the instruction is sent, there is a 50ms software delay inserted to allow for the tag response (the digest) to be buffered. The response shown in Figure 7.5 is 19 bytes long so the time required is as follows.

<table>
<thead>
<tr>
<th>Start</th>
<th>Digest data</th>
<th>CRC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 byte</td>
<td>16 bytes</td>
<td>2 bytes</td>
</tr>
</tbody>
</table>

**Figure 7.4 Get digest instruction**

**Figure 7.5 Digest reply message**
Response = 19 bytes x 8bits = 152 bits

19200bps = 19.2 bits per millisecond

152 bits ÷ 19.2 = 7.91ms (ideal transfer rate)

We show that approximately 60ms is needed for the microcontroller to read the digest data from the RW-210 module. The total time required to read a digest from the tag would be:

3.33ms + 50ms + 7.91ms = 61.24 ms

Next we look at the GUI display message for an authentication that passes as this is the longest single message sequence. Figure 5.10 shows the display beginning with “Calculated MD5 digest: ” and ending with the new line for the “system ready” prompt. The reader to host communication is set at 38400bps and the required time is shown;

163 characters x 8bits per character = 1304 bits

38400bps = 38.4 bits per millisecond

1304 bits ÷ 38.4 = 33.9ms

We show that approximately 34ms is needed for the SerialComm application to display the authentication status. The combined ideal communication times (62ms +
34ms = 96ms) are less than one tenth (0.100) of a second. This value is consistent with the actual test results we have observed.

This is comparable to the authorization time required when using an ATM machine to access one’s bank account. This is for a reader attached to a wireless modem, as described in Section 3.3 or local host compatible of RS-232 communication, however, there may be an additional time delay if the data must be sent over the internet. The internet routing calculations are not part of this thesis.

7.4. **System cost**

Our reader is not a production ready device, it is a prototype made from perforated substrate intended to be used for rough electronics work typically using through-hole components and hand soldered wires. To be a consumer product, the reader would require at minimum a dual layered printed circuit board (PCB) that has electrical traces in place of the hand soldered wires and use surface mount technology (SMT) components. The PCB cleans up the design and reduces the possibility of transient signals. The SMT components can reduce the cost of production as it can be automated. Also, certifications from the FCC and UL Labs are usually associated with a commercial product and must be in compliance. The design and certification costs are one-time expenses that would diminish when amortized over the production life of the product. The cost of our prototype is shown in Table 7.1
<table>
<thead>
<tr>
<th>Component</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATmega64</td>
<td>$12</td>
</tr>
<tr>
<td>MAX233CPE</td>
<td>$10</td>
</tr>
<tr>
<td>LEDs and misc. passive components</td>
<td>$15</td>
</tr>
<tr>
<td>RW-210 RFID module</td>
<td>$55</td>
</tr>
<tr>
<td>Prototype board</td>
<td>$4</td>
</tr>
<tr>
<td>QFP-to-DIP adaptor</td>
<td>$10</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>$106</strong></td>
</tr>
</tbody>
</table>

Table 7.1 Prototype cost

The cost of a commercial system would also be dependant on the communication features of this product. The most likely features would be an incorporated wireless radio transmitter or an embedded Ethernet module to increase the range of deployment beyond our testing configuration. Either option can be obtained as a daughter card and attached to the microcontrollers serial communication pins.

A commercial version would not use the prototyping board or the DIP adaptor, and the component costs would be much less when purchased in quantities of 1000 or more. A commercial product will also need to be enclosed to protect the components and require some form of retail packaging, which all adds to the product cost. If a deployed system was to use the HN-550 wireless radio, the units are readily available at major distributors and cost approximately $550 each. The radios were already in our inventory and were not a cost factor for our system.

The benefit of our system is that a population of 60 people can be serviced with a single reader. By utilizing this reader in lieu of a backend server, the maintenance costs of a server with its associated cast are mitigated. Due to the fact that our reader can interface with any RS-232 compatible device, there is no need for specific software. Our test system used a wireless radio eliminates the need for a network Ethernet
infrastructure, and may be deployed anywhere. Based on these criteria, it is reasonable to say that our standalone reader would cost less than a dedicated backend server used in the typical systems deployed today.
Chapter 8  Conclusion

8.1.  Closing remarks

Telemedicine is slowly gaining favor and is being introduced into health care facilities, but still faces criticism for security obstacles that need to be resolved. We showed that despite current RFID tag technology not being capable of real time encryption calculations, we can deploy a system that is able to conceal the patient’s identity.

In this thesis we set out to investigate a means of ensuring the confidentiality of a patient’s identity when using remote medical consultation referred to as telemedicine. We intended to show that patient identity verification was viable with RFID and that a typical backend server driven RFID system could be streamlined to perform the verification without a server.

Our RFID system is uses dual chain hashing establishing forward security in the authentication process between reader and tag. The authentication process is compartmentalized in the reader, and we do not require a back end server to provide the intelligent query to the tag as typical systems do. Our system communicates with any device that has a standard RS232 serial communications port and basic serial communications software eliminating the need for complex interfaces. Our system also reduces the risk of a backend server compromise by limiting the patient roster to 60 or less patients. In the event of a compromise, only the location to which the reader is assigned is affected, not the entire care facility. In this thesis, we have demonstrated that our embedded version of the RFID system is indeed viable and offers benefits to the
current backend server based systems being commissioned in care facilities adopting telemedicine.

8.2. Areas for future research

There are several areas for future research and development.

- The development of a computational tag would be a key addition to the proposed RFID authentication system. Computational capability would eliminate the described method of digest storage on the tag, thereby making simultaneous real time calculations possible on both reader and tag. That event will result in the verification process being invulnerable to attacks. Even if the data exchange was intercepted, the attacker would not be able to reverse the digest and discover the message.

- The integration of the Ethernet module or cell phone interface would give the RFID verification system the ability to be deployed anywhere the respective service is available. That greatly increases the deployment possibilities in modernized environments, but does not include all scenarios. The wireless radio tested solves our specific dilemma. A system that is capable of all three mediums would give the optimal deployment option, however the cost could be drastically increased. An investigation to implement this robust device would be of interest.

- Our system was intentionally developed to be capable of a simple interface so any device could communicate with it. The development of a comprehensive software application with user interfaces for the care provider and patient would make the system a better product offering.
Bibliography


