A Fraud-Prevention Framework for Software Defined Radio Mobile Devices

A Thesis
Presented to
The Academic Faculty

by

Alessandro Brawerman

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineering

Georgia Institute of Technology
August 2005

A Fraud-Prevention Framework for Software Defined Radio Mobile Devices

Approved by:

Professor John A. Copeland, Advisor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor Mary Ann Ingram
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor Henry Owen
School of Electrical and Computer Engineering
Georgia Institute of Technology

Date Approved: June 21, 2005

To my wife Celia, my parents and sister.

ACKNOWLEDGEMENTS

I am truly grateful to everyone who has helped me during this period of my life. It was a 4-year period of hard working that has finally paid-off. I have learned a lot not only in my research area and professional life, but also in my personal life. I have grown and matured, and now I am sure that I am ready to face any challenge that will come along.

First of all, I thank God for giving me this opportunity, as well as many others in my life, for giving me the necessary strength to overcome all the obstacles and for giving me light specially when I did not know what to do.

I would like to thank my advisor Prof. John A. Copeland for all his guidance and support, and the talks we had in his office and during our group meetings. His vision and experience were essential to me and inspired me to go beyond my own limits.

I would also like to thank Prof. Henry Owen, Prof. Mary Ann Ingram, Prof. Douglas Blough and Prof. Ellen Zegura for serving on my thesis committee. I specially thank Prof. Henry Owen, who accepted a last-minute invitation and gave me important feedback to improve my thesis. A special thanks also to Prof. Douglas Blough, who was my former advisor and gave me support during the first 2 years of my Ph.D.

I am also very grateful to my friends in the lab; they gave me important feedback during the presentation rehearsals of my papers, proposal and final defense.

Last but not least, I would like to thank my wife Celia, who gave up her life in Brazil to stay with me during this time. She gave me strength, comprehension and above all she gave me love when I most needed. I also thank my parents, Elias and Rosane, and my sister, Andressa, who always believed in me and gave me support even with the distance that separates us. Thanks to the four of you. You will always have a special place reserved in my heart for all eternity.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AC Attestation Credential

AC Eng Attestation Comparison Engine

ACP Attestation Credential Protocol

AM Eng Attestation Measurement Engine

AMPS Advanced Mobile Phone System

AK Attestation Key pair

AR Eng Attestation Report Engine

AS Eng Attestation Store Engine

Att(X) Attestation function = hash of X

AU Adaptive Unigram model

CA-EKDB Privacy CA's Database

CDMA Code Division Multiple Access

CFG Config file

CUP CFG Update Protocol

EK Endorsement Key

FPGA Field-Programmable Gate Array

GSM Global System for Mobile communication

J2ME Java 2 Micro Edition

KSSL KiloByte Secure Socket Layer protocol

LDDA Light Differential Download Algorithm

LSSL Light Secure Socket Layer protocol

Manuf SDR-MD's Manufacturer

PCs Personal Computers

PDAs Personal Digital Assistants

PII Pseudo-random Interface Identifier

PPM Prediction by Partial Matching model

Privacy CA Privacy Credential Authority

RA Regulatory Agency

R-CFGs Radio Configuration files

SDR Software Defined Radio

SDR-DM Software Defined Radio Device Manager

SDR-MDs Software Defined Radio Mobile Devices

SIM card Subscriber Identity Module card

SSL Secure Socket Layer protocol

TCG Trusted Computing Group

TMI Temporal Mobile Identifier

TPHP Tamper-Protected Hardware Package

TRCs Tamper Resistant Chips

UMTS Universal Mobile Telecommunications System

VoIP Voice over IP

WO Wireless Operator

WO-DB Wireless Operator's Database

SUMMARY

The superior reconfigurability of software defined radio mobile devices has made it one of the most promising technology on the wireless network and in the mobile communication industry.

The evolution from a static and rigid system to a highly dynamic environment, which offers many advantages over current systems, has been made possible thanks to the concepts of programmability and reconfigurability introduced by the software defined radio technology and the higher level of flexibility and openness of this technology's devices.

Clearly, the software defined radio mobile device's flexibility is a great advantage since the customer is able to use the same device in different parts of the world, with different wireless technologies.

Despite the advantages, there are still issues to be discussed regarding security. According to the Software Defined Radio Forum some of the concerns are the radio configuration download, storage and installation, user's privacy, and cloning.

To address the SDR Forum concerns a fraud-prevention framework is proposed. The framework is composed by new pieces of hardware, new modules and new protocols that together greatly enhance the overall security of software defined radio mobile devices and this new highly dynamic environment.

The framework offers security monitoring against malicious attacks and viruses that may affect the configuration data; protects sensitive information through the use of protected storage; creates and protects an identity for the system; employs a secure and efficient protocol for radio configuration download and update; and finally, establishes an anti-cloning scheme, which not only guarantees that no units can be cloned over the air but also elevates the level of difficulty to clone units if the attacker has physical access to those units. Even if cloned units exist, the anti-cloning scheme is able to identify them and deny any service.

CHAPTER I

INTRODUCTION

The latest advances in the mobile communication industry have led to the integration and intercommunication of existing and future networks together with radio access technologies. Mobile communications have been evolving from a static and rigid system to a highly dynamic environment that offers many advantages over current systems. For instance, it allows users to move seamlessly between different types of networks to optimize their quality of service or minimize the cost of using their mobile devices.

To improve telecommunication networks as we know them at present to systems with higher level of flexibility and openness it is necessary to change this rigid model by introducing the concepts of programmability and reconfigurability at various levels of network functionality.

One of the key elements to successfully achieve the benefits of having a highly dynamic environment is the availability of Software Defined Radio (SDR) technology and Software Defined Radio Mobile Devices (SDR-MDs) with reconfigurable protocol stacks and their equivalent on the network side.

1.1 Software Defined Radio and Software Defined Radio Mobile Devices

SDR [2] allows multiple radio standards to operate on common radio frequency hardware, thereby ensuring compatibility among legacy, current, and evolving wireless communication technologies.

SDR-MDs are capable of having their operation changed by dynamically loading Radio Configuration files (R-CFGs) over the air. With different R-CFGs the device can operate using different wireless communication technologies while having a single transceiver. A typical SDR-MD can manage communication via satellite, over different cellular technologies,

Voice over IP (VoIP), and operations over the Internet.

Despite several advantages there are still issues to be discussed regarding security. According to the SDR Forum [27] some of the concerns are the R-CFG download, storage and installation; user's privacy, i.e., protection of the user's identity, location, and communication with other devices; and finally, SDR-MD cloning, i.e, illegally using services that are billed to someone else's device. The search for efficient techniques for R-CFG differential download is also a relevant issue.

1.2 Research Targets

To address the SDR Forum concerns and greatly enhance the overall security of SDR-MDs a fraud-prevention framework is proposed. A list of the main objectives and key contributions of this framework follows below:

- To offer security monitoring against malicious attacks and viruses that may affect the configuration data
- To protect sensitive information through the use of protected storage
- To create and protect an identity for the system
- To download and update R-CFGs in a secure and efficient manner
- To establish an anti-cloning scheme that makes the device itself aware of cloning and guarantees that no cloned unit is able to use the network services

Practical experiments using Java 2 Micro Edition (J2ME) [21] and proofs that analyze the correctness of the fraud-prevention framework are also presented.

1.3 Thesis Outline

The rest of this thesis is arranged as follows. Chapter 2 discusses relevant related work. In Chapter 3, basic definitions and the entities involved in the fraud-prevention framework are presented. Chapter 4 specifies a light secure socket layer protocol. In Chapter 5, security checks necessary to install a new R-CFG are proposed. Chapter 6 presents a light differential

download algorithm responsible to compress R-CFGs. In Chapter 7 the anti-cloning scheme is proposed. Chapter 8 presents a secure and efficient protocol for R-CFG download and update. This is followed by the conclusion in Chapter 9.

CHAPTER II

RELATED WORK

Research work has been done for each of the SDR Forum concerns described in Chapter 1; however, no published work has developed a solution that encompasses more than one of the concerns at once. This chapter is divided according to the SDR Forum concerns. For each section, some of the relevant related research is presented.

2.1 R-CFG Download, Storage and Installation

In [23], the authors discuss a model for securing R-CFG download and installation that involves the use of secret device keys and signatures. All security operations take place within tamper-proof hardware that also contains the programmable components of the transceiver.

Their approach provides good security for the radio software that lies within the tamper-proof hardware, but leads to some drawbacks such as the use of non-standard security methods, lack of a means for third-party vendors to provide R-CFGs, and, most importantly, lack of a means for securing radio software that resides outside the tamper-proof hardware.

2.2 User's Privacy

Some efforts, called privacy extension to Mobile IPv6, deal with user's privacy. The basic idea of these efforts is to replace the MAC address of a mobile device with a random one, called a Temporal Mobile Identifier (TMI) [8] or Pseudo-random Interface Identifier (PII) [11].

In those schemes, personal mobile location privacy control relies on either the home administration, the foreign administration, or both. Moreover, the home administration is required to share some secrets with the foreign administration to prevent eavesdroppers from having any knowledge about binding users temporal identifiers and real identifiers.

These efforts cannot completely control mobile location privacy by a mobile user since the administration can associate any identifier (PII or TMI) with the corresponding real ID of the mobile device.

Another interesting scheme is proposed in [25]. The authors propose an authorized-anonymous-ID-based scheme. The scheme effectively eliminates the need for a trusted server or administration. The key point is a cryptographic technique called blind signature, which is used to generate an authorized anonymous ID that replaces the real ID of an authorized mobile device. With authorized anonymous IDs, the authors design an architecture capable of achieving complete personal control over location privacy while maintaining the authentication function required by the administration.

2.3 SDR-MD Cloning

The Advanced Mobile Phone System (AMPS) [26, 29] is the analog mobile phone system standard, introduced in the Americas during the early 1980s. Despite the fact it was a great advance in its time, the AMPS presented several security flaws, and multiple copies of cloned mobile stations were created with little difficulty.

The Global System for Mobile communication (GSM) [16, 26, 29] is a globally accepted standard for digital cellular communication. The GSM authentication framework relies on special cryptographic codes to authenticate customers and bill them appropriately. A personalized smartcard, called Subscriber Identity Module card (SIM card), stores a secret key that is used to authenticate the customer; knowledge of the key is sufficient to make calls billed to that customer.

The SIM card is easily removable so that the user can use other cellphones. The drawback is that someone who has physical access to the SIM card can copy the information to another card, thereby, cloning the authentication information of the user.

Cloning the SIM card is a relevant flaw; however, a much more serious flaw was discovered. In [33] it is shown that the cryptographic codes used for authentication are not strong enough to resist attacks. To exploit this vulnerability, an individual would interact repeatedly with the SIM card to learn the secret key and would then be able to clone the

phone without having to clone the SIM card. Although it was considered that the attacker had physical access to the SIM card, it was mentioned that over the air attacks are possible, making cloning on GSM cellphones a more serious threat.

The Universal Mobile Telecommunications System (UMTS) [26, 29] is an open air-interface standard for third generation wireless telecommunications. It provides higher data rates and fixes several security flaws encountered in the GSM standard. Despite several advantages that the UMTS standard provides, it also stores vital information in the SIM card. Thus, like the GSM, someone might be able to copy the authentication information from one SIM card to another.

Another drawback concerns the KASUMI block cipher, which is at the core of the integrity and confidentiality mechanisms in the UMTS network. Hardware implementations are required to use at most 10000 gates and must achieve encryption rates in the order of 2Mbps (maximum data rate). Thus, a considerable effort must be performed in order to implement a high performance hardware component that carries out the operations of the KASUMI block cipher.

As a final remark, UMTS devices are not capable of reconfiguring their radio parameters via software. Thus, dual mode or tri-mode expensive cell phones are necessary to guarantee backward compatibility with other standards.

Simpler schemes that only detect cloned units and do not try to prevent cloning have also been proposed. They can be found in [14, 24].

2.4 Differential Download

Rsync [32] takes a different approach to differential download that allows a client to request changes to a file from the server without requiring the server to maintain old versions. The server calculates the differences on the fly. This is a drawback since extra time is necessary. Besides, Rsync requires a large number of operations on the client side. Thus, it would present low performance if employed by SDR-MDs, which are by nature constrained and use a low bandwidth network.

The Xdelta algorithm [22] is based on the idea of block fingerprints introduced by Rsync.

This algorithm also uses adler32 and MD5 checksums to generate fingerprints. However, unlike the Rsync, it requires that the server have all the existent versions of the requested file. Thus, the differences can be generated off-line, a priori. An advantage of Xdelta is that it uses a split encoding that separates the sequence of instructions from the data output. The performance of Xdelta is also unsatisfactory for constrained SDR-MDs since its logic depends on the use of computer-intensive operations executed by a couple of Linux libraries, such as glib and zlib.

2.5 Trusted Computing Group

The Trusted Computing Group (TCG) [30] is an industry standards body comprising computer and device manufacturers, software vendors, and others with an interest in enhancing the security of the computing environment across multiple platforms and devices.

The TCG claims that it will develop and promote open industry standard specifications for trusted computing hardware building blocks and software interfaces across multiple platforms, including Personal Computers (PCs), servers, Personal Digital Assistants (PDAs), and digital phones.

So far the TCG has only presented specification for the PC environment [31]. Some of the benefits include more secure local data storage, a lower risk of identity theft, and the deployment of more secure systems and solutions based on open industry standards.

Despite the fact that the TCG specification for the PC does point out and solve several security flaws, this specification would not achieve a satisfactory performance if employed by constrained SDR-MDs.

CHAPTER III

A FRAUD-PREVENTION FRAMEWORK FOR SOFTWARE DEFINED RADIO MOBILE DEVICES

The superior reconfigurability of the software defined radio technology has made it one of the most promising technology on the wireless network and in the communication industry. Clearly, SDR-MDs' flexibility is a great advantage since the customer is able to use the same device in different parts of the world, with different wireless technologies.

Despite several advantages, there are still issues to be discussed regarding security. According to the SDR Forum some of the concerns are:

- To secure the radio configuration file download, storage and installation
- To protect the user's identity, location, and communication with other devices
- To deny network services to cloned units

To address the SDR Forum concerns and greatly enhance the overall security of SDR-MDs, a fraud-prevention framework is proposed. The proposed framework main contributions are:

- To offer security monitoring against malicious attacks and viruses that may affect the configuration data
- To protect sensitive information through the use of protected storage
- To create and protect an identity for the system
- To download and update R-CFGs in a secure and efficient manner
- To establish an anti-cloning scheme that makes the device itself aware of cloning and that, not only denies services and identify cloned units, but also guarantees that no

units can be cloned over the air. The scheme also elevates the level of difficulty to clone units when the attacker has physical access to the devices.

The fraud-prevention framework is composed of new pieces of hardware, new modules, and new protocols. Figure 1 depicts the design of the framework. The dashed squares are the main contributions of this work.

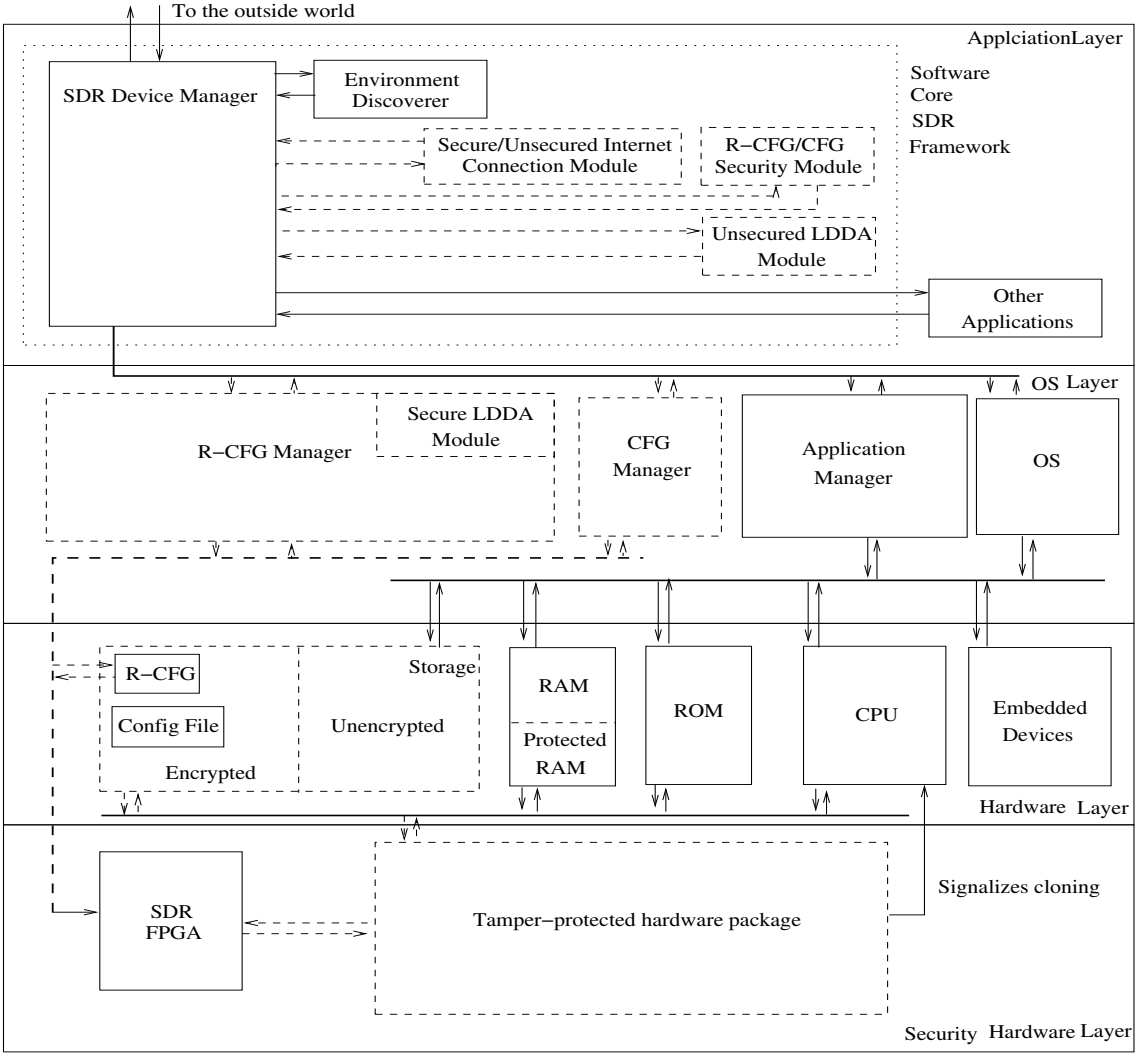

Figure 1: The design of the fraud-prevention framework.

Note that the Software Defined Radio Device Manager (SDR-DM) is responsible for managing all the communication with the outside world and for requesting the services of each module when needed. Also, the Environment Discoverer module is responsible for detecting which wireless communication technologies are available in the current SDR-MD's

environment. This module is assumed to be present in the Software Core SDR Framework and is outside the scope of this work.

The R-CFG manager is responsible for managing the R-CFG files currently stored in the device and the R-CFG currently installed. It also informs the SDR-DM when a different R-CFG is needed. The CFG manager is responsible for managing the Config file (CFG). The CFG is provided by the Wireless Operator (WO) and is used to set the device's phone number. Other modules are discussed in separate chapters as part of the new protocols and the anti-cloning scheme.

It is important to mention that the modules may work together to form the entire framework or they may be used independent as an individual piece to perform some action.

3.1 Basic Definitions

This section presents basic definitions and components that are employed in the fraud-prevention framework. The nomenclature used to specify the framework is presented in Table 1.

The size of each key employed in the framework may vary according to the technology used. The smallest key size must be equal to 128 bits, on the other hand, there is no limit in the upper bound. As the devices computing capabilities grow, so can the key size without degrading overall performance.

3.2 Entities

There are five entities that participate and have different responsibilities within the fraud-prevention framework. The list below presents the entities as well as their responsibilities.

1. SDR-MD's Manufacturer (Manuf)
    - Produces the SDR-MD
    - Generates the R-CFGs and delta-sets to update old versions
    - Generates the SDR-MD's EK and informs the Privacy CA about the EK
    - Calculates and stores the Att(EK) in the SDR-MD

Table 1: Basic definitions.

C	a 48-bit nonce.
$K_Y\{C\}$	C is cryptographically transformed, somehow, with a key Y.
$MD(Z)$	hash of Z.
$[C]_{Alice}$	C is transformed using the private key of Alice.
$\{C\}_{Alice}$	C is transformed using the public key of Alice.
Attestation	It is used to check the integrity status of a certain component. It is defined as the function Att(X), which results in the hash of component X.
Attestation Key pair (AK)	It is used to obtain the attestation credential. Composed by the attestation private key (AK_{priv}) and public key (AK_{pub}).
Attestation Credential (AC)	It is used to identify the SDR-MD. It is signed by the Privacy Credential Authority ($PrivacyCA$) and it is presented whenever the user tries to utilize network services. $AC = [hash(AK_{pub})]_{PrivacyCA}$.
Endorsement Key (EK)	It is used to uniquely identify the SDR-MD. It is never disclosed by the device.
R-CFG file	It is a structured binary file that is applied to reconfigure the Field-Programmable Gate Array (FPGA) of an SDR device. It is formed by FPGA commands that are able to change radio parameters, power output, frequency range, etc.
CFG file	It is used to set up the phone number of the SDR-MD. It is signed by the Wireless Operator. $CFG = [Phone\#]_{WO}$.
Temporary State	An SDR-MD that cannot identify itself to the network, thus it is not ready to be used. It does not have the AC, the CFG or the R-CFG.
Valid State	An SDR-MD ready to be used. It already has valid AC, CFG, and R-CFG.

2. Regulatory Agency (RA)

   - Tests, approves and licenses the R-CFG. Basically, the RA tests the R-CFG in the specific hardware to ensure that the device does not cause interference or function out of its defined spectrum, as defined in [12].

3. Wireless Operator (WO)

   - Sells the SDR-MD
   - Provides network services
   - Generates the CFG
   - Authenticates the SDR-MD to use the network services

- Detects cloned SDR-MDs

4. Privacy Credential Authority (Privacy CA)

   - Provides the SDR-MD with an AK pair, the AC, and the WO public key

5. SDR-MD

   - Utilizes the network services
   - Downloads R-CFGs and CFGs files
   - Verifies the R-CFG before installing it
   - Detects if it is a cloned or valid unit

Figure 2 depicts the relationship between entities. The Manufacturer sends the EK to the Privacy CA, and the R-CFG for approval to the Regulatory Agency. The Regulatory Agency tests the R-CFG and either approves or denies it, and sends the result to the Manufacturer. After obtaining an approval, the Manufacturer sends the approved R-CFG to the SDR-MD. Then, the SDR-MD obtains the AK pair, the AC, and the WO's public key from the Privacy CA. Finally, the SDR-MD obtains the CFG from the WO and it is ready to use the network services.

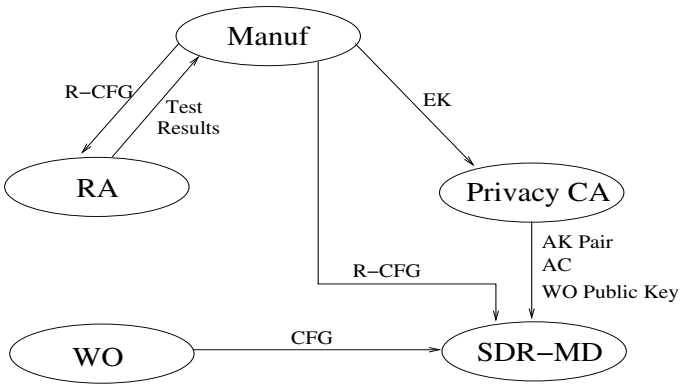

Figure 2: The relationship between entities.

The fraud-prevention framework was originally published in [7]. A more complete version is subjected to be submitted and can also be found in [6].

CHAPTER IV

A LIGHT SECURE SOCKET LAYER PROTOCOL FOR SOFTWARE DEFINED RADIO MOBILE DEVICES

An SDR-MD is usually compared to a radio PC that can host different air interfaces applications and the major focus is on the access system. Experiences from the PC platform have shown that the user welcomes the opportunity to install and use third-party software on his/her system for personalization issues [10].

Several of the third-party software, including R-CFGs, that may be downloaded and installed by the SDR-MD have proprietary information. In such cases, it is necessary to secure the server/SDR-MD connection during the software download to protect the secret information from being eavesdropped by other users.

To that end, it is proposed a protocol to securely connect servers and SDR-MDs, called Light Secure Socket Layer protocol (LSSL). The LSSL protocol is a lightweight version of the Secure Socket Layer protocol (SSL) [15]. Thus, the LSSL is more suitable for SDR-MDs operating under low-capabilities, low-bandwidth, and error-prone wireless links.

Experiments, presented later in this chapter, show that the time to perform the initial setup when employing the SSL protocol is unnecessarily long. The initial setup comprises the time to establish connection, negotiate the cipher suite, and generate the session key.

The LSSL takes up less bandwidth and reduces the computational load on the client side, transferring the majority of the computational work, mathematical and encryption operations, to the server side. In this fashion the LSSL is able to improve performance of the initial setup and to reduce the overall execution time.

The LSSL, which was originally published in [3], is implemented in the secure/unsecured Internet connection module of the framework (Figure 1). Both the LSSL and the SSL are provided to obtain a secure connection. The protocol provided to obtain an unsecured connection is the HTTP.

4.1 The SSL Protocol: Overview

The SSL protocol provides bulk encryption, end point authentication, and integrity protection of application data usually over unsecured-public networks. The protocol requires a reliable, bi-directional, byte-stream service, which is typically provided by the TCP.

It employs symmetric key algorithms to provide encryption and authentication services. The keys for these services are established by the handshake protocol, which uses public/private key algorithms to create a session key between the SSL client and the server. The session key is known only by the server and the client and it is used to encrypt the messages exchanged by them.

The SSL is very flexible and can accommodate a variety of algorithms for key agreement, for instance RSA; for encryption, RC5; and for hashing, MD5. During the protocol, the client and the server exchange messages to agree which of each algorithm will be employed. The combination of these algorithms is called cipher suites.

Also, during the protocol, the client generates a random number that is further used to calculate the session key. It is the client's responsibility to encrypt that random number with the server's public key and send it to the server, so that the server can also use that number to calculate the session key.

The SSL protocol allows mutual authentication, i.e., both client and server can authenticate each other. Usually, only the server maintains a certificate and the client is authenticated through the use of a password. The full SSL handshake protocol is shown on Figure 3A.

4.2 The LSSL Protocol: Specification

SDR-MDs are usually constrained devices, i.e., they typically do not have much processor power, memory, or long battery lifetime. The goal with the LSSL is to have a simple yet efficient protocol that is more suitable to securely connect SDR constrained devices and servers. The LSSL performs similar tasks to the SSL in providing basic cryptographic functions such as encryption, hashing, and authentication. However, the new protocol redesigns the SSL, reducing the client computational load and leaving the computational

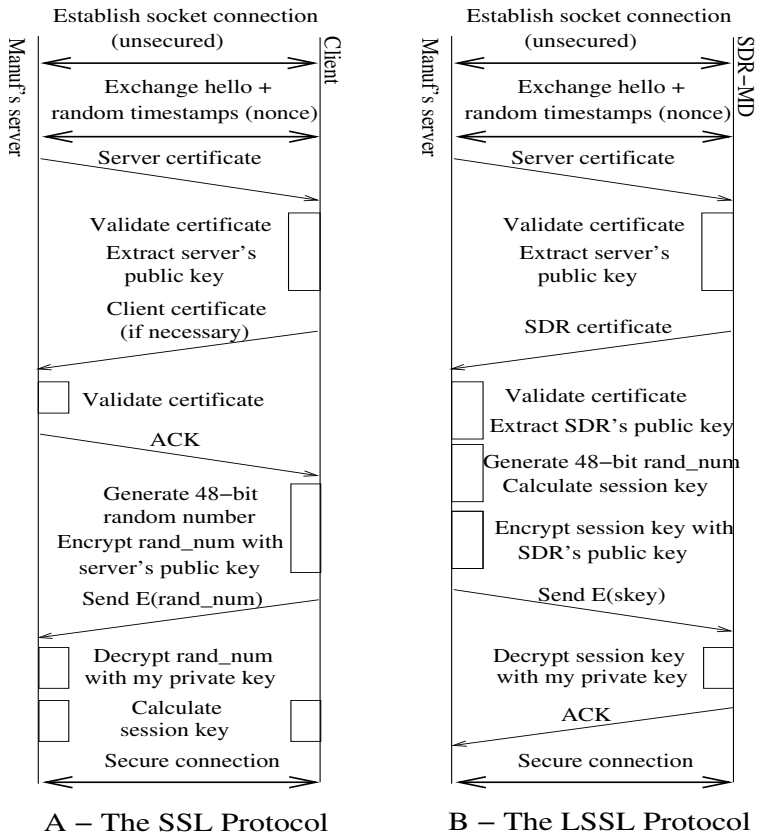

Figure 3: The SSL and LSSL protocols.

intensive operations on the server side.

The LSSL is designed to use any end-point authentication mechanism combined with minimal message exchanges without compromising the level of security. It works with a chosen subset of available methods. The default algorithms are RSA for public/private key exchanges with key lengths up to 2048-bits, RC5 for bulk encryption and MD5 for message digesting. Certificates accepted are X.509 certificates with no restriction on the chain length.

Figure 3B shows the LSSL protocol. At first, an unsecured connection is established and the hello messages are exchanged. Notice that random timestamps (nonces) are also exchanged, so as in the SSL, the LSSL is safe regarding replay attacks.

Next, the LSSL performs mutual authentication employing certificates for both sides. After that, the server (not the client as in the SSL) performs operations to obtain the session key. Finally, the server sends the session key to the client. Notice that, different from the

SSL, the client does not need to generate and encrypt the random number, nor does it need to calculate the session key. The only operation the client needs to perform is to decrypt the session key it has received from the server.

4.3 LSSL Related Work

Another attempt to develop a lightweight SSL protocol, called KiloByte Secure Socket Layer protocol (KSSL), is presented in [17]. The KSSL is more restrictive than the LSSL, since it offers even fewer cipher suites, and does not perform client side authentication, which is required by the SDR Forum. It also does not remove the computational intensive operations from the client side (the LSSL's main idea). Since the KSSL does not present a great improvement over the SSL, the authors seem to have discontinued this project.

4.4 LSSL Practical Experiments

In this section, two experiments involving the LSSL are presented. In the first experiment, the LSSL is executed with different encryption/decryption algorithms and different session key sizes. In the second experiment, the LSSL is compared with the SSL regarding performance. Each experiment was executed 50 times and the final results were averaged.

The setup for these experiments is depicted in Figure 4. The client connects to the server through a wireless firewall router. The three entities on the setup are:

1. The manufacturer server: a Dell Inspiron 5100 Pentium IV - 2.6 GHz with 256MB RAM and Linux OS. It provides HTTP and HTTPS services by the use of the Apache Software [1].

2. The client: a Sharp Zaurus PDA CL-760 with CPU speed of 400 MHz, 64MB SDRAM, Linux OS and J2ME support.

3. The router: a Netgear 108Mbps wireless firewall router. Model WGT624.

The graph in Figure 5 shows the results of the first experiment in which the LSSL performance, when using several encryption/decryption algorithms, is measured. This experiment does not intend to compare the performance of the LSSL with those algorithms,

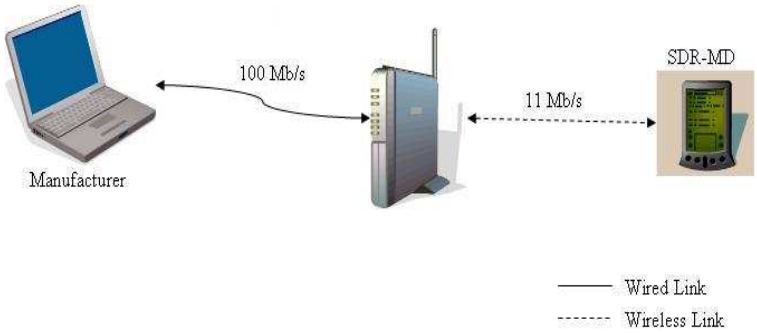

Figure 4: LSSL experiment setup.

since the key size differs for each algorithm; rather the goal of this experiment is to show that the LSSL works with the most common encryption/decryption algorithms.

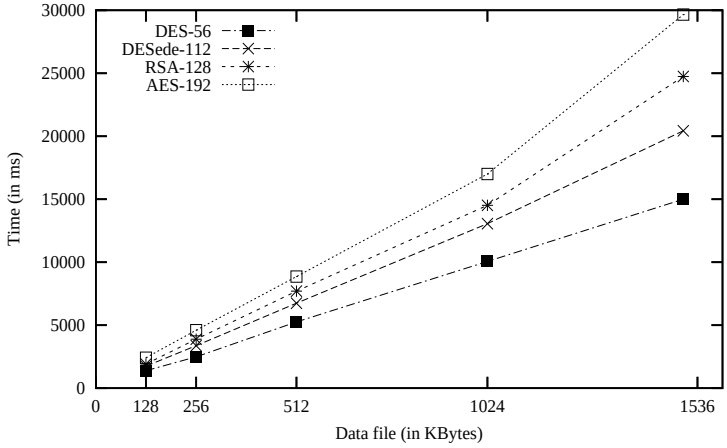

Figure 5: LSSL and common cryptographic algorithms.

The performance measurements presented in this experiment, for each cryptographic algorithm, comprises: the time to perform mutual authentication, the time for the server to perform the session key related operations, the time for the server to encrypt the data file, the time for the client to download the encrypted data file, and finally, the time for the client to decrypt the data file.

As it was expected, because of the session key size, the LSSL presents its best performance, regarding completion time, when using the DES algorithm with a session key of 56 bits and the worst performance when using the AES with a session key size of 192 bits. For instance, in the case that the data file size is equal to 1MB, it takes about 10 seconds to perform the whole process when using DES and about 17.5 seconds when using AES.

The second experiment is more interesting. It compares the LSSL and the SSL performance when transferring data files of different sizes, over the air, from the manufacturer's server to the SDR-MD. The AES algorithm is employed to create the 128-bit session key and encrypt the data file. The graph in Figure 6 shows the comparison results of two breakpoints.

Break1, which represents the initial setup, comprises the time to establish an unsecured connection, exchange hello messages and timestamps, negotiate the cipher suite, mutual authentication, and operations to generate the session key. Notice that the LSSL completes break1 much faster. This is because the operations to generate the session key are performed by the server, which is a much more powerful machine.

Break2 comprises the time to encrypt the data file by the server, the time the SDR-MD takes to download the E_{skey}(data_file), and to decrypt the data file. These are similar in both protocols, with a larger delay when using the LSSL to transfer large files.

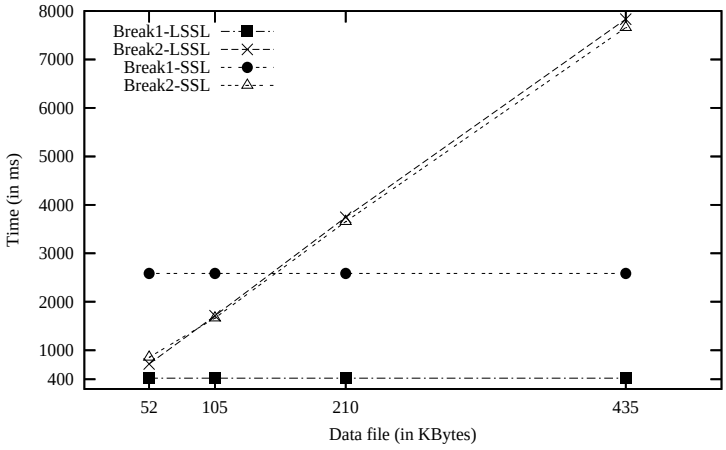

Figure 6: LSSL versus SSL. Break comparison.

The graph in Figure 7 shows the LSSL versus the SSL total time comparison, i.e., the time the SDR-MD takes to complete Break1 + Break2. Notice that the process is faster when using the LSSL for all cases. In fact, the difference in performance is constant and approximately 2.1 seconds. This is because the savings are obtained during the initial setup (Break1), which is independent of the amount of data to be transmitted.

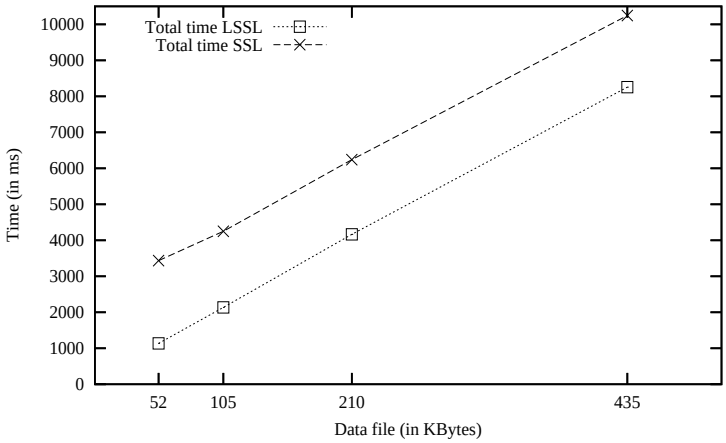

Figure 7: LSSL versus SSL. Total time comparison.

4.5 Considerations Regarding LSSL, SSL and Security

The SSL protocol was chosen to be ported to SDR communications mainly because it is the most used security communications protocol on the Internet and it has proven to be safe against most kinds of attacks.

By porting the well-known SSL protocol to a more suitable protocol for SDR-MDs, one can get the impression that the SSL would be more secure, or that the SSL would safeguard more kind of attacks.

However, by the way the LSSL is structured, the same kind of attacks prevented by the SSL is also prevented by the LSSL. Replay attacks are prevented by employing nonces; masquerade attacks are prevented by performing mutual authentication; access to the session key is given only to the server, which has calculated the key, and to that specific client, since the session key is encrypted with the client's public key; and finally, a secure connection is obtained, since only the server and that specific client know the session key.

On the other hand, the SSL can be considered more complete since it presents more cipher suite options; the client has much more processor power, so key sizes can be bigger and cryptographic operations would not take that long to be completed; the client has much more storage space, so it can store its key pair plus anything that is needed with no problem at all; it can transmit more messages for flow and error control without compromising overall performance; and finally, the session key is not transmitted over the connection.

In terms of cipher suites, the LSSL presents the most common ones, so it is unlikely that the LSSL would not work for a pair of client/server that the SSL would. Regarding key sizes, the LSSL performs operations with keys of up 2048 bits with minimal delay. The tradeoff of having bigger keys, thus more security, against smaller keys, thus faster completion time, will always occur. Finally, the LSSL transmits the session key over the connection, but it is encrypted with the client's public key, thus, only that specific client is able to decrypt the session key.

CHAPTER V

SECURING THE DOWNLOAD OF RADIO CONFIGURATION FILES

One of the main advantages of SDR-MDs is their ability to move between different types of networks to optimize their quality of service. Suppose an SDR-MD device that currently works in Code Division Multiple Access (CDMA) mode. If the device is hand over to a GSM network and it does not have a GSM R-CFG installed, it can contact its manufacturer, download the GSM R-CFG, and install it. The scheme is depicted in Figure 8.

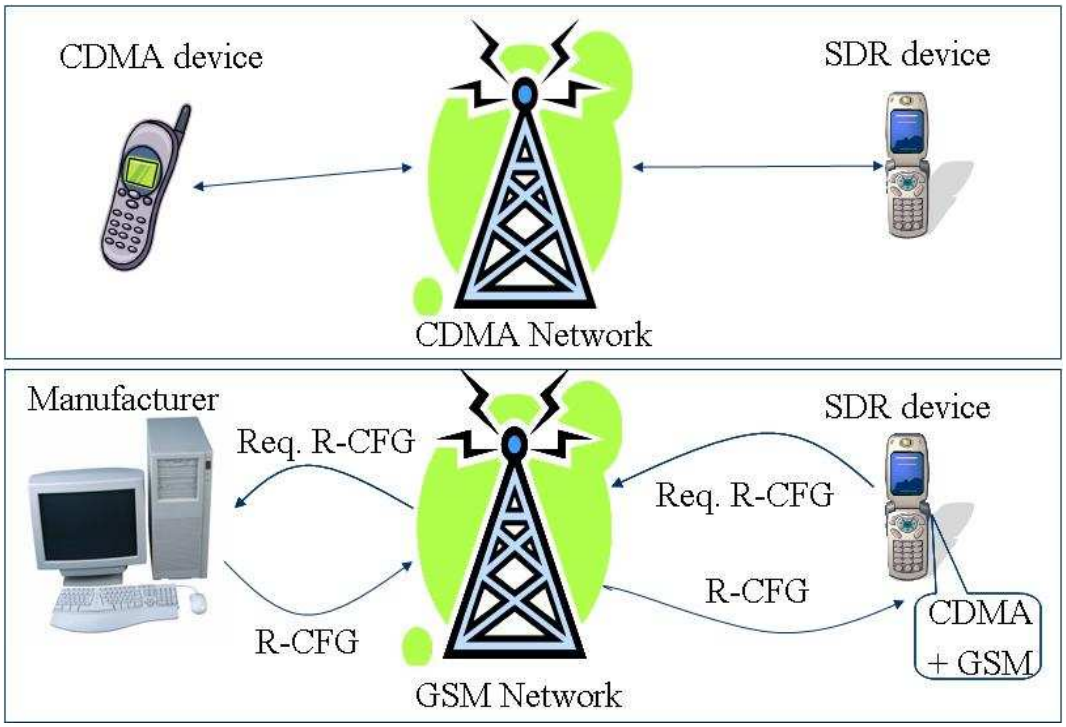

Figure 8: The R-CFG download.

Although the scheme works correctly and the SDR-MD is indeed able to reconfigure its radio parameters, it is not guaranteed that a valid R-CFG, i.e., an R-CFG that has been approved by the RA, has been downloaded. Therefore, security safeguards must be implemented to prevent attacks that seek to install invalid R-CFGs or malicious data.

The SDR Forum defines a set of security requirements for wireless software download. Based on those guidelines, the R-CFG/CFG security module describes the steps of mutual authentication, public/private key mechanisms for data encryption and decryption, and fingerprint calculations to check data integrity.

The specification of the R-CFG/CFG module was originally published in [3].

5.1 The R-CFG/CFG Security Module

Whenever a manufacturer generates a new R-CFG, it has to send the R-CFG to be approved and licensed by the RA. This is called R-CFG validation.

5.1.1 R-CFG Validation

A public-private key mechanism is employed to perform R-CFG validation. The manufacturer sends to the RA a combination of a header, which contains manufacturer, model, serial number range, and possibly some other information; the new R-CFG; and the hardware in which the R-CFG is to be tested and used.

The RA installs the R-CFG in the specified device and tests the device's behavior. If no malfunction is observed, the RA approves the R-CFG and assigns it a license number. During the test, the RA computes $h = MD(header||R-CFG)$. The value h is then signed with the RA's private key, $[h]_{RA}$. Figure 9 depicts the signing step. The signed hash value, $[h]_{RA}$, is sent back to the manufacturer along with the assigned license number.

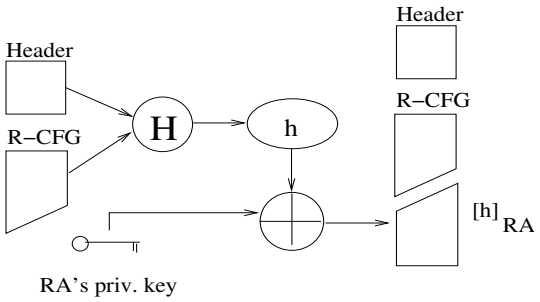

Figure 9: R-CFG validation.

Once the R-CFG has been licensed, signed, and placed on a server, the SDR-MDs can contact the server at any time to download the combination of header, R-CFG, and $[h]_{RA}$.

5.1.2 R-CFG Verification and Data Integrity Check

Whenever a new R-CFG has been downloaded, R-CFG verification is necessary to guarantee that the R-CFG has been approved and properly signed by the RA. The verification step also tests whether the R-CFG is appropriate for the device (Figure 10A).

However, to guarantee that the R-CFG has not been modified after being approved, the following steps are performed:

1. The SDR-MD calculates a new hash value *h'* by inputting the received header and R-CFG into the same hash function used when the R-CFG was signed;
2. The SDR-MD decrypts the received $[h]_{RA}$, with the RA's public key, to obtain h;
3. The SDR-MD compares h and *h'*: if $h = h'$, the received R-CFG is accepted. However, if $h \neq h'$, the SDR-MD rejects the R-CFG.

Figure 10B shows the data integrity check.

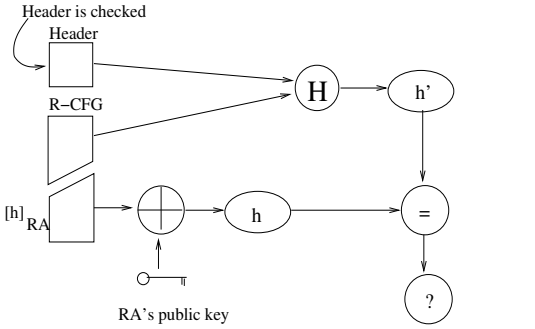

Figure 10: R-CGF verification and data integrity check.

5.2 Security Checks Experiment

The goal in this section is to present an experiment that measures the time it takes to perform R-CFG verification and data integrity check.

The experiment setup is the same as in Figure 4, i.e., an SDR-MD, in this case a Sharp Zaurus PDA SL-5600 with CPU speed of 400 MHz, 32MB SDRAM, Linux OS and J2ME support, connects to the manufacturer's server through a wireless firewall router.

An invalid R-CFG, i.e., R-CFG that has not been authorized by the RA or that has been modified after the RA has signed it, is downloaded. The SHA-1 algorithm is used to calculate the fingerprint and to perform the data integrity check. RSA-128 bit keys are used as the RA's public/private key pair.

The graph in Figure 11 shows the results for this experiment. As expected, the larger the R-CFG is, the longer it takes to perform the security checks. For instance, it takes about 600 milliseconds to verify a 128KB R-CFG, about 1.08 seconds to verify a 1MB R-CFG, and about 2.35 seconds to verify a 2MB R-CFG.

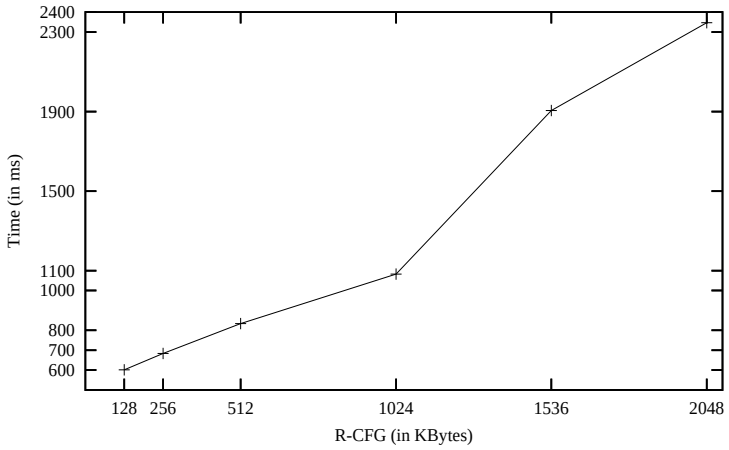

Figure 11: Time to identify invalid R-CFGs.

These results show that a compression method should be used to avoid performing the security checks in large files, avoid disconnecting the service during the new R-CFG installation, and minimize transmission errors. Since R-CFGs are executable files that are loaded in the FPGA of an SDR-MD, standard compression algorithms, such as GZip, do not work well. Thus, a new algorithm to compress R-CFG files is proposed in the next chapter.

CHAPTER VI

R-CFG DIFFERENTIAL DOWNLOAD

R-CFG files for SDR-MDs can be downloaded over the air, allowing these devices to support multi-mode functionality using a single transceiver. However, SDR devices are usually constrained devices, thus, storing several R-CFGs, one for each mode available, might not be the best solution.

Another solution would be to download a new R-CFG version, to update the current R-CFG or to exchange modes, only when necessary. The drawback is that the wireless link is also constrained and downloading the entire R-CFG could take some time. For instance, it takes about 2.2 seconds to transfer a 1MB R-CFG and 4.5 seconds to transfer a 2MB R-CFG.

To achieve a better solution, avoid service disconnection, and minimize transmission errors during the new R-CFG installation, the use of differential download is proposed. Since R-CFGs are similar in their basis, i.e., they usually have a similar set of commands, there is no need to download the entire R-CFG when exchanging to a different mode or updating to a new version of the same mode. Downloading the differences between the installed R-CFG and the one to be used is more efficient.

This chapter presents a new algorithm for differential download, referred to as Light Differential Download Algorithm (LDDA). The LDDA is responsible for calculating a delta set between an old R-CFG and a new R-CFG. The delta set is the difference between those R-CFGs. With this application, the SDR device downloads the much smaller delta set instead of the entire R-CFG.

The LDDA is the first differential download algorithm specifically designed for SDR constrained devices. It presents several new features that make it useful for updating R-CFGs within the same mode, exchanging to a different mode, or updating any software by differential download.

Some of the novel ideas of the LDDA are optimization tailored for R-CFG files, integration of delta set generation and data integrity check, efficient instruction logic, elimination of redundancy on the data file, simpler and smaller delta sets, and independence of OS platform. Practical experiments are performed to demonstrate the feasibility and superior performance of the LDDA when compared with other approaches.

The LDDA was originally published in [4].

6.1 LDDA Basics

Consider an old R-CFG currently installed and maintained by a certain SDR-MD. Suppose the old R-CFG is subject to be exchanged to a new R-CFG, which might be or not of the same mode. Instead of downloading the entire new R-CFG, the idea is to download only a difference result set, called a delta set, which represents the difference between the old R-CFG and the new R-CFG.

The LDDA delta set comprises two files: delta file and data file. The data file contains the new data that appears in the new R-CFG and is not present in the old R-CFG, i.e., the new data to be inserted. The delta file is a list of commands used to exchange or update the SDR-MD's R-CFG. The commands indicate whether to copy data from the old R-CFG or to copy new data from the data file.

Since the volume of the delta set is significantly smaller than that of the raw new R-CFG, the advantage is obvious, downloading the delta set is a much faster and more efficient process than downloading the raw R-CFG. The smaller the delta set is, the faster it is to transport it electronically, the fewer chances for transmission errors to occur, and the less it will cost to update the software.

To carry out the update at the SDR device, the device's manufacturer should execute the LDDA to generate the delta set and make it available for the download. The process involves:

- The LDDA server side, which compares the old R-CFG and the new R-CFG, and generates the delta set
- Communication between the manufacturer's server and the SDR-MD

- The LDDA client side, which incorporates the differences into the old R-CFG, thereby generating the desired new R-CFG

Figure 12 depicts the process. The manufacturer generates the delta set. The SDR device connects to the manufacturer and downloads the most up-to-date delta set for the requested mode. The SDR-MD applies the changes into the old R-CFG, generating the new R-CFG.

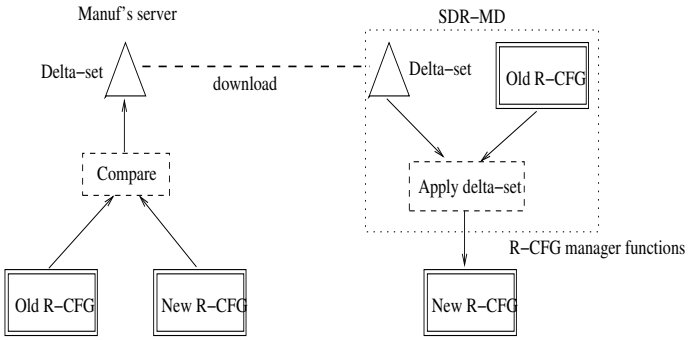

Figure 12: SDR R-CFG differential download.

The LDDA is not responsible for installing the R-CFG on the client side. It assemblies the R-CFG and makes it available to the R-CFG manager, so that it can be installed later.

6.2 LDDA Specification

The nomenclature used to specify the algorithm is as follows:

Old R-CFG - the current R-CFG installed in the SDR device. Located on both server and client.

New R-CFG - file to replace the old R-CFG. Together with the old R-CFG, it is used to generate the delta set on the server side. Located only on the server.

Updated R-CFG - file that is the same as the new R-CFG, but located on the client side. It is generated using the delta set and the old R-CFG.

Data file - file that contains the new data to be inserted. Located on the server and downloaded by the client.

Delta file - file containing a list of instructions to generate the updated R-CFG from the old R-CFG and data file. Located on the server and downloaded by the client. The following instructions can be found within a delta file:

1. copy X-Y: copies a block of commands from block X up to block Y, from the old R-CFG to the updated R-CFG. This instruction is used when there is a command or a block of commands that is the same in the old R-CFG and in the new R-CFG.
2. insert X-Y: copies a block of commands from block X up to block Y, from the data file to the updated R-CFG. This data does not exist in the old R-CFG, only in the new R-CFG.

In reality, there are only copy instructions in the LDDA. The insert instruction is a copy instruction that refers to the data file instead of the old R-CFG. This instruction is left as insert to facilitate comprehension.

The LDDA is optimized for SDR R-CFG download, it treats the R-CFG as a sequence of commands and not just an unstructured binary file. Because of the R-CFG structure, the LDDA presents features that other delta compression algorithms do not, e.g. efficient instruction logic and elimination of redundancy in the data file.

6.2.1 Efficient Instruction Logic

Different from previous algorithms, the LDDA presents efficient instruction logic, i.e., it tries to group in a single instruction as many Field-Programmable Gate Array (FPGA) commands as possible. Suppose that the last instruction in the delta file is: insert 5, which means copy from the data file the fifth FPGA command. Now, suppose the current instruction informs to copy the sixth FPGA command from the data file. The LDDA will edit the last instruction on the delta file to also contain the sixth command. Thus, the final instruction is: $insert\ 5-6$.

The efficient instruction logic makes the final delta file smaller and easier to be interpreted by the client and it has shown to improve overall performance. Table 2 illustrates how the efficient instruction logic works. Note that the current instruction is to insert

command 6. Since the last instruction on the delta file is an insert instruction and the command present on that instruction (command 5) immediately precedes command 6 on the new R-CFG, there is no need to include a new insert instruction for command 6, only editing the last instruction to include command 6 is enough.

Note also that, in the LDDA delta set, the copy instruction is copying block 1 through 4, while in other algorithms it would be necessary one instruction for each command.

Table 2: Efficient instruction logic.

Current Instruction	LDDA Delta File	Other Algorithms Delta File
Insert 6	Copy 1-4	Copy 1
	Insert 5-6	Copy 2
		Copy 3
		Copy 4
		Insert 5
		Insert 6

6.2.2 Delta Set Creation

The delta set creation occurs only on the server side. The server maintains all existent versions of each R-CFG, from the oldest one up to the most up-to-date. It also maintains the delta set between each old version and the most up-to-date. Thus, when the client needs a certain delta set, the server has already generated it and it is ready to be downloaded.

It is important to mention that since the delta sets are generated a priori, and the servers are not constrained devices, the time required to generate delta sets is not a big concern. This is a disadvantage of the LDDA; the delta creation usually takes more time whn employing the LDDA than when employing the Xdelta, for example. However, the LDDA concentrates on creating a clean and easy to interpret delta set. Therefore, when the client reads the delta file, it does not have to do many operations, i.e., the LDDA reduces the workload on the client side.

Together with the delta file, the server also creates a delta header that contains: name and version of the old R-CFG; name and version of the new R-CFG; fingerprint of the new R-CFG.

The name and version of the old R-CFG ensure that the client will use this delta set

for the correct old R-CFG file. The name and version of the new R-CFG generate the correct new R-CFG name on the client, and finally, the fingerprint is used to prevent the installation of an incorrect R-CFG. It is a common practice to calculate the fingerprint after the server has generated the delta set. However, with the LDDA there is no need to analyze the delta-set again, the fingerprint is generated concurrently with the delta set.

The pseudo-code to create the delta set is presented below:

```
delta_set_creation() {
    open old R-CFG;
    while (!EOF) {
        read X = command or block of commands;
        calculate fingerprint of X;
        input fingerprint and X index in a hash table;
    }
    close old R-CFG;
    open new R-CFG;
    while (!EOF) {
        read Y = command or block of commands;
        calculate fingerprint of Y;
        accumulate fingerprint to new R-CFG fingerprint;
        if (fingerprint is on hash table)
            efficient_instruction_logic(copy, Y);
        else efficient_instruction_logic(insert, Y);
   }
   close new R-CFG;
}
```

A hash table is used to discover whether an FPGA command occurs in both, the old R-CFG and the new R-CFG. Table 3 presents the hash table structure. The fingerprint field stores the fingerprint of a command or block. In this example, suppose the fingerprint is an hexadecimal number of 16 bits. The command index field shows the index of that command within the old R-CFG. For instance, the fingerprint for the second command and the forty-eighth command is $ABCD$, which means they are actually the same command.

The LDDA builds the hash table by mapping fingerprint values to their offsets for blocks of size s in the source version. Hash collisions are unlike to happen for large blocks, such as blocks of 128 bits. This is the default value in the LDDA. For smaller blocks hash

Table 3: Hash table structure.

Fingerprint	Command Index
ABCD	2 48
1234	190
1AD4	117

collisions can occur with a small probability, but, more important, the overall performance is decreased when employing small blocks.

Tables 4 and 5 show how to create the delta set having the old and new R-CFG. The example is in high-level language (the fingerprints in the parenthesis were not actually calculated). From the old R-CFG the hash table is created. By comparing the new R-CFG and the hash table, the delta set is created. The delta file contains the list of commands and the data file contains the new data to be inserted.

Table 4: Old and new R-CFGs.

Old R-CFG	New R-CFG
X = 3 (0014)	X = 3 (0014)
Y = 5 (0023)	Y = 5 (0023)
S = X + Y (0139)	Z = 2 (0012)
Y = Y - 1 (0123)	S = X + Y (0139)
S = X + Y (0139)	X = Y - X (0146)
	Y = S + Z (0158)
	X = Y - X (0146)

Table 5: Delta set creation.

Hash Table	
Fingerprint	Command
0014	1
0023	2
0139	3 5
0123	4

Data File	Z = 2 X = Y - X Y = S + Z
Delta File	Copy 1-2 Insert 1 Copy 3 Insert 2-3 Insert 2

6.2.3 Eliminating Redundancy on the Data File

Different from other algorithms, the LDDA is able to eliminate command redundancy in the data file. If an FPGA command to be inserted (new data) occurs more than once in

the new R-CFG, it will only appear once in the data file, thus greatly reducing the overall delta set size. To correctly assemble the updated R-CFG, the delta file will contain as many references to that command as it occurs in the new R-CFG.

Table 6 shows how the elimination of redundancy works in the LDDA. Notice that commands 1 and 4 are the same, as well as commands 2 and 5. The elimination of redundancy logic notices the code redundancy, and includes those commands only once in the data file. The references to the data file are there in the delta file, and when following the instruction list, the updated R-CFG is assembled correctly.

Table 6: Eliminating code redundancy.

New R-CFG	... X = Z + Y (1) Y = Z * X (2) Z = Z - 1 (3) X = Z + Y (4) Y = Z * X (5) ...
LDDA - Data File	X = Z + Y (1) Y = Z * X (2) Z = Z - 1 (3)
LDDA - Delta File	Insert 1-3 Insert 1-2

Xdelta - Delta File	
New Data	X = Z + Y (1) Y = Z * X (2) Z = Z - 1 (3) X = Z + Y (4) Y = Z * X (5)
Instruction List	Insert 1 Insert 2 Insert 3 Insert 4 Insert 5

Because the LDDA introduces the efficient instruction logic on the delta file and the elimination of redundancy on the data file the LDDA's delta set is smaller than the Xdelta's delta file, which is the file that contains the difference set in the Xdelta. Note that in Table 6 the LDDA's delta file contains only 2 instructions, while the Xdelta's delta file contains 5 instructions. Also, the LDDA's data file contains only 3 new commands, while the Xdelta's delta file contains 5 new commands. Therefore, the LDDA's delta set is transferred faster than the Xdelta's delta file and chances to occur transmission errors are smaller.

6.2.4 The Update Phase

The update phase occurs in the client side and since SDR-MDs are constrained devices, it has to be as simple and as efficient as possible.

In the LDDA, after having downloaded the appropriate delta set, the client only needs

to interpret the delta file and copy blocks from either the old R-CFG or the data file to the updated R-CFG. This technique has shown to improve overall performance since, different from other differential download algorithms, there is only one type of instruction on the delta file.

Table 7 shows an example of the update phase; it follows the example in Table 5. From the delta set plus old R-CFG, the updated R-CFG is generated. As it can be seen, the updated R-CFG contains commands 1 and 2 from the old R-CFG, command 1 from the data file, followed by command 3 from the old R-CFG, and finally, commands 2 and 3 from the data file.

Table 7: The update phase.

Delta File	Data File	Old R-CFG
Copy 1-2	Z = 2 (1)	X = 3 (1)
Insert 1	X = Y - X (2)	Y = 5
Copy 3	Y = S + Z (3)	S = X + Y (3)
Insert 2-3		Y = Y - 1 (4)
Insert 2		S = X + Y (5)

Updated R-CFG = New R-CFG
X = 3
Y = 5
Z = 2
S = X + Y
X = Y - X
Y = S + Z
X = Y - X

The pseudo-code for the update phase is presented bellow:

```
update_phase() {
    open delta file and data file;
    open old R-CFG; //get name & version from the header
    create updated R-CFG; //get name & version from the header
    for each instruction on delta file {
        if (instruction == copy)
            copy blocks from old R-CFG;
        else if (instruction == insert)
            copy blocks from data file;
        accumulate block fingerprint;
    }
    close data file and old R-CFG;
```

```
    compare final fingerprint with the one in the header;
    if (they are the same) completion;
    else error;
    close delta file and updated R-CFG;
}
```

6.3 The LDDA's Adavantages

The advantages of LDDA are listed below:

1. Optimized for R-CFG download: the LDDA correctly interprets the R-CFG, i.e., each block of bytes read is one FPGA command or block of commands. Because of this fact, the LDDA is able to find how many times and where in the flow one command occurs in that specific R-CFG. This improves update time, since the data file is compressed to its minimal.

2. Data integrity check: the LDDA takes into account the fact that MD5 values are already being calculated to compare the new R-CFG and old R-CFG, so instead of calculating the MD5 value for the whole new R-CFG, it accumulates the individual MD5 values of each command. With the LDDA there is no need to analyze the new R-CFG again after the delta creation or the update phase has been completed.

3. Efficient instruction logic: the LDDA discovers when subsequent FPGA commands are being copied and creates only one instruction for those commands.

4. Delta set: the LDDA delta set is divided into 2 files to avoid reading unnecessary data during the update phase, making it easier and faster.

5. Elimination of redundancy on the data file: new data that appears more than once in the new R-CFG will appear only once in the data file. With this feature the LDDA's delta set is usually smaller than the ones generated by other algorithms.

6. Update phase: the LDDA updates the R-CFG faster since its delta file is simpler to be interpreted by a constrained device. $Update\_phase = download\_time + Updated\_R-CFG_creation_time$.

7. OS platform independent: the LDDA is fully implemented using J2ME, so it is OS platform independent.

8. Updating other softwares: the LDDA can be used to update any kind of software, as long as the server has an old and a new version of the same software. However, the LDDA is optimized to be employed with R-CFGs, thus its performance when updating other softwares is not a concern.

6.4 LDDA - Practical Experiments

In this section the LDDA is compared with the Xdelta. The results obtained with the experiments show that the techniques employed by the LDDA generate better performance, regarding completion time.

For the first 5 experiments Figure 13 depicts the setup. A Pentium 4 2.6GHz with 728MB RAM and Linux OS is used as the manufacturer's server and a Pentium 4 2.6GHz with 256MB RAM and Linux OS is used as the client. The network connection is established through a Netgear 108Mbps wireless firewall router.

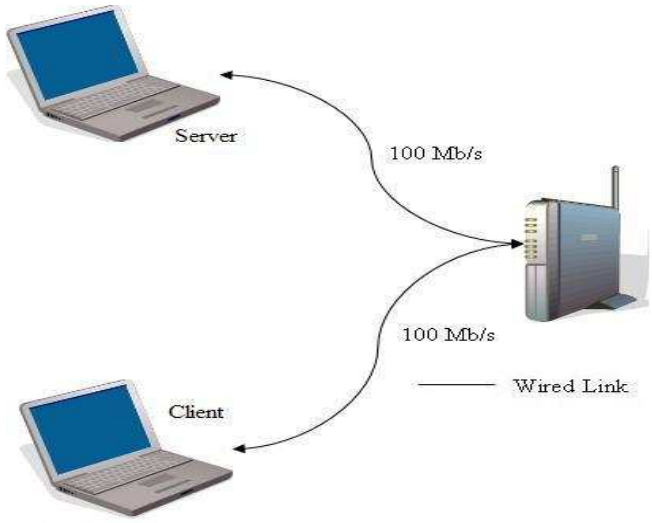

Figure 13: Differential download setup experiment.

Because the Xdelta requires computationally intensive operations executed by the Linux libraries, glib, and zlib, it has not been ported to SDR devices yet. Thus, in the first 4 experiments, no SDR devices have been used. Even though this setup is not an optimized

environment for the LDDA, the LDDA has already proven to be faster than the Xdelta. The LDDA performance in a constrained SDR-MD is evaluated in experiment 6.

6.4.1 Experiment 1: Integrating Delta Set and Integrity Check

In this experiment, the technique of integrating the delta set creation and new R-CFG data integrity check is compared against the technique of calculating the fingerprint after having created the delta set.

The graph in Figure 14 shows the comparison between the LDDA integrated scheme against LDDA using a non-integrated scheme. For this experiment, a 1MB R-CFG base (old R-CFG) is used and new data is inserted in the new R-CFG. Therefore, fields like 1024 + 128 mean that the old R-CFG has $1024KB = 1MB$ and 128KB more are inserted in the new R-CFG.

As it can be noticed, the LDDA technique of integrating the delta creation and the data integrity check performs better. This is a small improvement by itself, but it will ultimately improve the overall algorithm's performance.

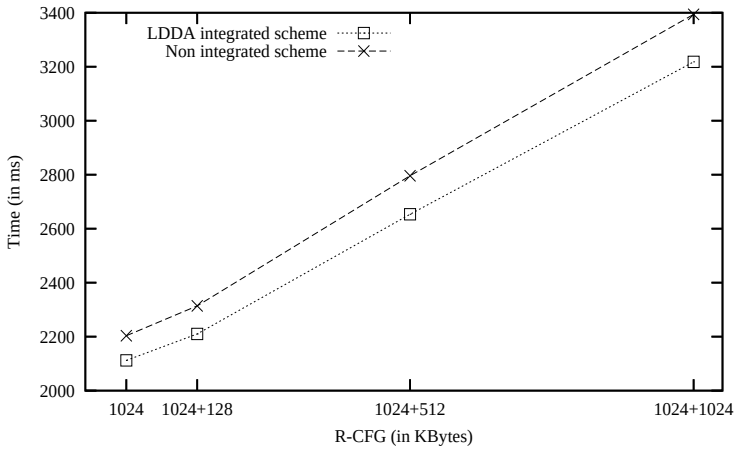

Figure 14: Integrated scheme vs. non-integrated scheme. Old R-CFG = 1024KB.

6.4.2 Experiment 2: Delta Set Size Comparison

In this experiment, the size of the delta sets created when using the LDDA and the Xdelta are compared. The results show, as mentioned before, that with the elimination of redundancy logic, the LDDA is able to create smaller delta sets, therefore saving time to transfer

and complete the update process.

The graph in Figure 15 shows the delta set sizes (in Kbytes) that are generated when using the LDDA with no code redundancy (LDDA 0), the LDDA with 10% code redundancy (LDDA 10), the LDDA with 20% code redundancy (LDDA 20), the LDDA with 30% code redundancy (LDDA 30), and finally, the Xdelta.

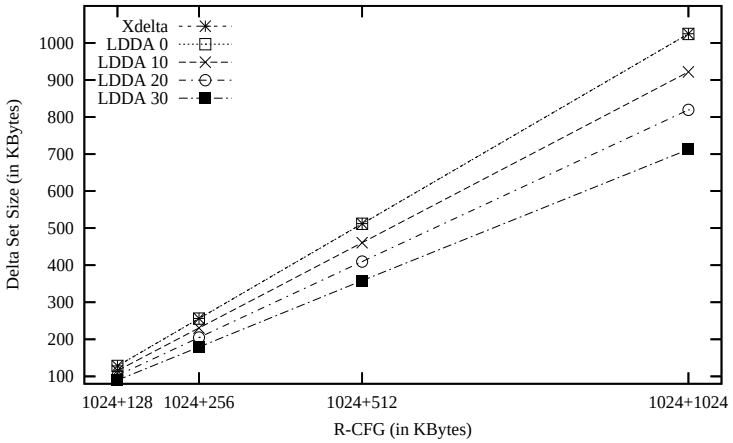

Figure 15: Delta set sizes.

In this graph, as in all following experiments, points like "1024 + 512" don't necessary mean that the delta set size is equal to 512KB. The meaning of "1024 + 512" is that the old R-CFG has $1024KB = 1MB$ and 512KB more are new data.

As it can be seen, by using the efficient logic instruction and the code redundancy logic features, the LDDA creates smaller delta sets in all cases, and the more the code is redundant, the smaller the delta set is.

6.4.3 Experiment 3: Update Phase Performance

In this experiment, the goal is to show the time it takes for the update phase to complete, i.e., the time to transfer the delta set is not computed. The graph in Figure 16 shows the time it takes for the client to generate the updated R-CFG when using the LDDA and the Xdelta.

It can be seen that in all cases the LDDA presents better performance, about 50% faster to execute the update phase. This is due to the facts that the LDDA is optimized for SDR downloads, so it interprets the R-CFG as a list of FPGA commands.

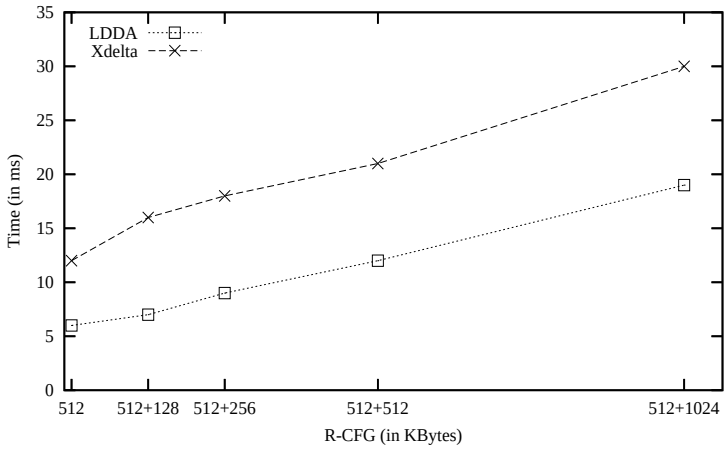

Figure 16: Comparison of total time to update R-CFG. Old R-CFG = 512Kbytes.

6.4.4 Experiment 4: LDDA vs. Xdelta

In this experiment, the LDDA is again compared with the Xdelta, but the goal here is to show the benefits of the elimination of redundancy logic, and the total time to generate the updated R-CFG on the SDR device (time to transfer delta set over the wireless connection + update phase time).

Six different tests are executed in this experiment. First, the time to transfer the entire new R-CFG is computed. Second, the time to execute the LDDA with no code redundancy, i.e., there is not a single FPGA command repeated within the new data, is computed. In the third, 10% of the new data is code redundancy. For the fourth, 20% of the new data is code redundancy. In the fifth, 30% of the new data is code redundancy. Finally, the time to execute the Xdelta is computed.

The graph in Figure 17 shows the results when transferring the entire new R-CFG and when executing the LDDA (with no code redundancy). Notice that the more up-to-date the current R-CFG version in the SDR is, the better the LDDA performs. For instance, in the case that 128KB are included in the new R-CFG (1024+128 case) the LDDA performs about 9 times faster, while in the case that 1MB is included (1024 + 1024 case), the LDDA performs about 2 times faster. This is because the older the R-CFG on the SDR device, the more new data has to be transmitted over the wireless connection. The bottleneck in this case is the network, not the SDR-MD.

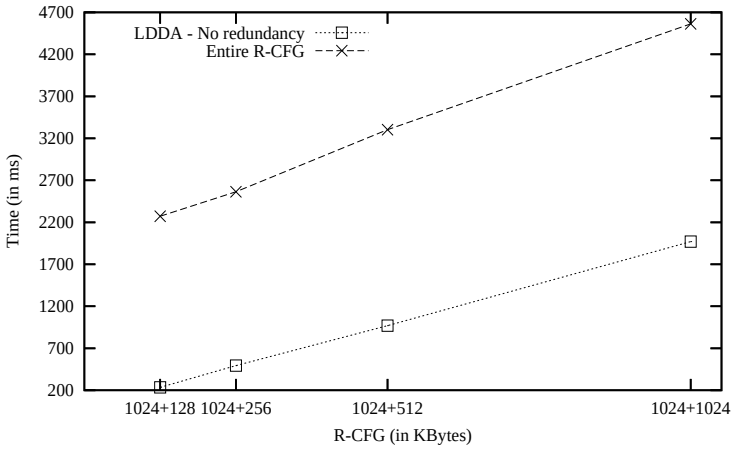

Figure 17: Entire R-CFG against using LDDA.

The graph in Figure 18 shows the results when executing Xdelta and LDDA with the code redundancy feature. The case of 1024 + 128 is not shown in this graph for readability purposes. It can be seen, as expected, that in all cases the LDDA with 30% code redundancy presents better performance. With the elimination of redundancy logic, the LDDA is able to create smaller delta sets, therefore the time to transfer and complete the whole update process is even faster than when using Xdelta. A 10% to 25% improvement is achieved by the LDDA, when completing the whole process with no code redundancy, and 30% improvement is achieved if 30% of the new data is redundant.

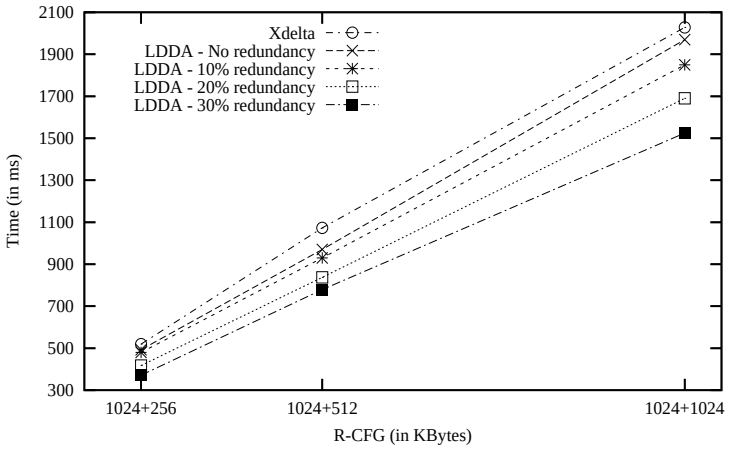

Figure 18: Using the elimination of redundancy logic.

6.4.5 Experiment 5: Compression Experiment

The goal in this experiment is to show that the delta set can be further compressed using binary compression algorithms; however, with the available algorithms it is more costly to compress the delta set, transfer the compressed file over the connection, decompress the delta set and finally update the R-CFG on the SDR-MD than simply executing the LDDA with no further compression.

The following common algorithms/models are employed to further compress R-CFGs delta sets: Gzip [18], LZ [34], arithmetic coding based on the Adaptive Unigram model (AU) [13], and based on the Prediction by Partial Matching model (PPM) [9].

Gzip and LZ are not efficient for R-CFG delta sets. They do not compress R-CFG delta sets very well, and in fact, Gzip generates a larger file due to information that is included in its header. On the other hand, the AU and the PPM model do compress R-CFGs delta sets. However, due to the intensive computational operations when executing those models, there is no real gain on performance.

Figure 19 shows the difference on performance when comparing the LDDA without code redundancy, the AU model, and the PPM model. For the LDDA, the time to transfer the delta set and update the R-CFG in the device is computed, and for the AU and the PPM only the time to decompress the delta set is computed. Notice that the time to decompress the delta set is much higher than the time to execute the LDDA. Therefore, those compression algorithms are not suitable for SDR constrained devices. The development of a light binary compression algorithm, capable of decompressing the delta set in a constrained device, adding minimal delay, is a topic of future work.

6.4.6 Experiment 6: A Complete Experiment in a Constrained Environment

The goal in this experiment is to evaluate the performance of the LDDA in a constrained environment. The experiment setup is similar to the one depicted in Figure 4. It comprises an SDR-MD, in this case a Sharp Zaurus PDA SL-5600 with CPU speed of 400 MHz, 32MB SDRAM, Linux OS and J2ME support, connected through an 11Mbps wireless link, to a Pentium 4 2.6GHz server with 256MB RAM providing HTTP services.

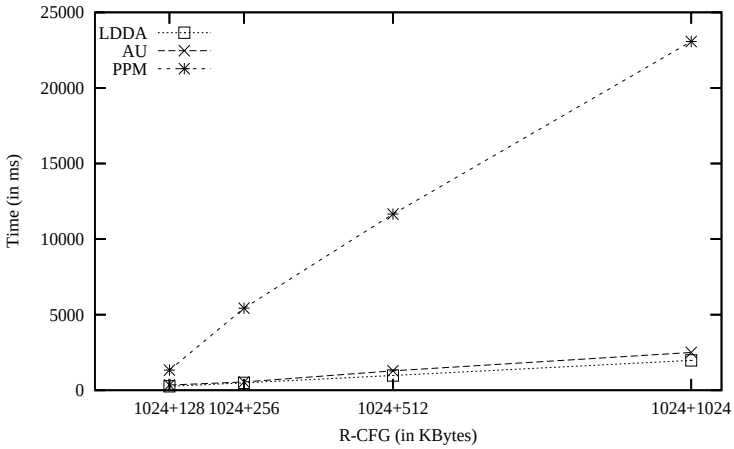

Figure 19: Decompressing the delta set.

The results in Figure 20 compare the LDDA against the method of transferring the entire raw R-CFG, both in a constrained and in a non-constrained environment. Note that the gain with the LDDA varies from 25% to 50%. This improvement in performance may prevent the SDR-MD from being disconnected during the R-CFG download.

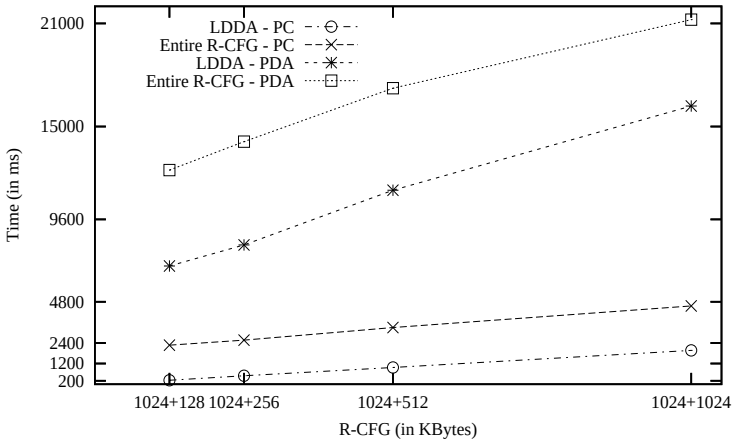

Figure 20: Constrained environment.

CHAPTER VII

THE ANTI-CLONING SCHEME

One of the more dangerous threats in SDR wireless communication is cloning. SDR-MD cloning is considered a federal crime. According to [28], telecommunication fraud losses are estimated at more than a billion dollars yearly. A large amount of this loss is due to cloning.

Besides illegal billing, cloned units increase the competition of shared resources, which increases network congestion and degrades network services. Furthermore, the impact of overload traffic from cloned units is unpredictable. Thus, the estimation of traffic patterns is imprecise for network planning.

In this chapter an anti-cloning scheme for software defined radio mobile devices is proposed. The anti-cloning scheme is designed to provide a core set of hardware and software technologies that establish the basis for a wireless network environment free of cloned units.

Unlike other schemes, in the proposed anti-cloning scheme, the SDR-MD is aware of cloning, i.e., it is able to identify if it has been cloned and take the necessary steps to block the use of the network services. As a security measure backup, the Wireless Operator is also responsible for detecting if a cloned SDR-MD is trying to use the network services.

Since SDR-MDs operate over different wireless communication technologies, another advantage of the proposed scheme is that it is independent of technology, working well for different cellular technologies, satellite networks and the Internet.

To obtain an SDR-MD that is aware of cloning, a tamper-protected hardware package and new protocols are defined. The tamper-protected hardware package stores vital information, such as the SDR-MD unique identifier and the SDR-MD attestation credential. Attestation engines, which verify the integrity status of the SDR-MD's elements, are also permanently encoded in the tamper-protected hardware package.

The protocols to be discussed define the steps of obtaining, validating and storing an attestation credential, and obtaining, validating, storing and installing a Config file (CFG),

which sets the phone number of the SDR-MD. Cloning aware procedures for both the SDR-MD and the WO are also defined. Finally, experiments and proofs that analyze correctness of the protocols and the framework are presented.

The anti-cloning scheme presented in this chapter was originally published in [5], and can also be found in [7, 6].

7.1 Tamper-Protected Hardware Package (TPHP)

The TPHP must be physically protected from tampering. This includes physically binding it to the other physical parts of the SDR-MD such that it cannot be easily disassembled and transferred to other devices. These mechanisms are intended to resist tampering.

Tamper evidence measures are to be employed. Such measures enable detection of tampering upon physical inspection. The package must limit pin probing and EMR scanning. Similar tamper-protected hardware are the trusted platform module of [30] and the Intel wireless trusted platform processor [20].

The TPHP is composed by two Tamper Resistant Chips (TRCs): $TRC1$, which is read only, and $TRC2$, which is read/write. The TRC1 contains the EK, the attestation engines responsible for measuring, reporting and comparing integrity values, and a specialized hardware to generate 48-bit random numbers. The TRC2 contains the attestation engine responsible for storing integrity values and protected non-volatile memory to store the necessary keys.

Figure 21 depicts the components of the TPHP as it comes from the Manufacturer. Notice that the TPHP comes from the Manufacturer with the RA's public key already stored.

The attestation engines are divided into the Attestation Measurement Engine (AM Eng), Attestation Store Engine (AS Eng), Attestation Report Engine (AR Eng), and Attestation Comparison Engine (AC Eng). Table 8 presents the functions of each attestation engine.

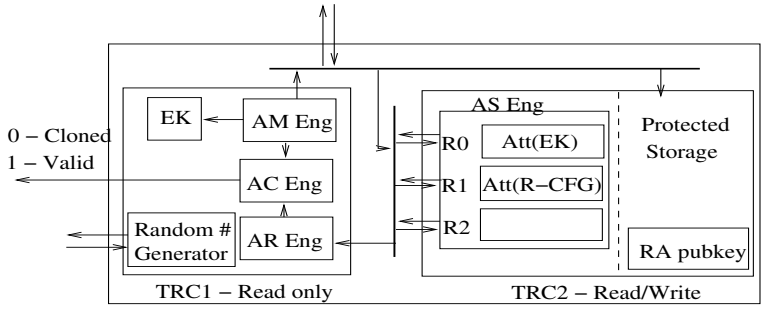

Figure 21: The tamper-protected hardware package in an invalid state.

Table 8: Attestation engines and functions.

AM Eng	Measures Att(EK), Att(R-CFG) and Att(CFG), and writes the results into R0, R1 and R2.
AS Eng	Stores the Att(EK) in register 0 ($R0$), the Att(R-CFG) in register 1 ($R1$) and the Att(CFG) in register 2 ($R2$).
AR Eng	Reads and reports the values of the registers.
AC Eng	Compares the values of R0, R1 and R2, reported by the AR Eng, with the values measured by the AM Eng.

7.2 SDR-MD - Entering a Valid State

The SDR-MD comes from the Manufacturer in an invalid state, i.e., it does not have the AC, therefore, it cannot identify itself to the network. After obtaining the AC, the SDR-MD enters a temporary state, i.e., it is able to prove its identity, however, it does not have a phone number yet, it does not have the CFG file installed. After obtaining the CFG, the SDR-MD finally reaches a valid state. It is able to identify itself and use the network services.

Figure 22 depicts the transition states that the SDR-MD has to go through in order to reach the valid state. Note that anytime after the SDR-MD has reached the valid state, it may need a new R-CFG file or a new CFG file. While obtaining any of those files, the SDR-MD goes to a temporary state. With the new data locally stored, the security checks are executed and the SDR-MD goes back to the valid state.

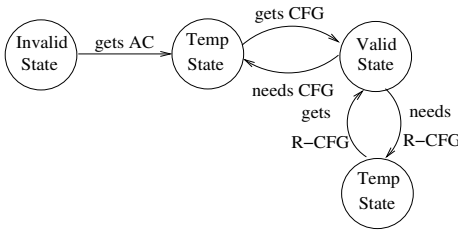

Figure 22: Transition states of an SDR-MD.

7.2.1 The Attestation Credential Protocol - ACP

To obtain a valid AC, the SDR-MD has to execute the Attestation Credential Protocol (ACP) depicted in Figure 23. The ACP is a communication process between the SDR-MD and the Privacy CA and it is executed only one time per each EK. The ACP is transparent to the final user.

Whenever the Manufacturer generates a new EK, it informs the Privacy CA, in a safe way, about that EK. The Privacy CA, like the Manufacturer, maintains a database of all available EKs, named CA-EKDB, indexed by MD(EK). This database has all information that the Privacy CA needs to know about each SDR-MD produced and links each SDR-MD to its AC.

The ACP steps are defined as follows: first, the SDR-MD contacts the Privacy CA and sends the value $R0 = Att(EK)$. The Privacy CA looks for a matching MD(EK) in the CA-EKDB. If it finds a match, the Privacy CA obtains the EK of that unit and acknowledges the unit. If no equivalent MD(EK) is found, either the Manufacturer failed to inform the Privacy CA about this unit or this is an invalid EK. Thus, the Privacy CA does not provide an AC to the unit.

Second, the Privacy CA generates an AK pair and the unit authenticates the Privacy CA. The unit generates a nonce C and sends it to the Privacy CA encrypted by the EK. The Privacy CA obtains C and sends it back along with an encrypted message containing the AK pair. Upon receiving the message, the unit verifies C, authenticating the Privacy CA.

Third, after authenticating the Privacy CA, the unit obtains the AK pair and acknowledges the Privacy CA. The Privacy CA then generates the $AC = [AK_{pub}]_{PrivacyCA}$ and

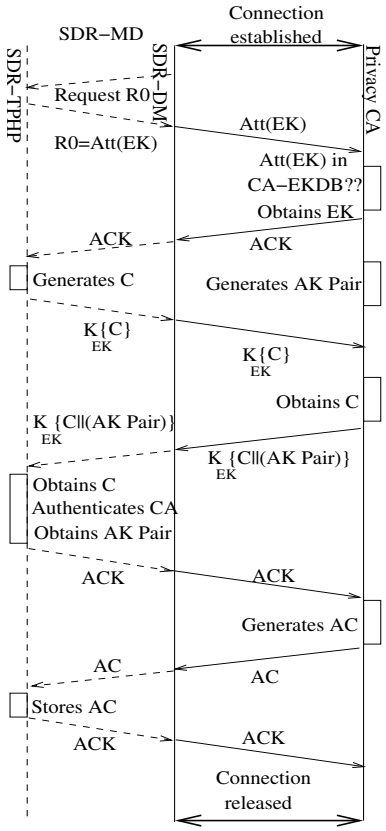

Figure 23: Attestation credential protocol.

sends it, encrypted by the AK_{pub} to the unit. The unit receives the AC, decrypts it, and stores it in its TPHP. After that, the connection is finally released.

7.2.2 The CFG Update Protocol - CUP

After obtaining the AC, the final step to enter the valid state is to have the SDR-MD executing the CFG Update Protocol (CUP) to obtain a valid CFG. This protocol is executed whenever the unit needs a new phone number and it is also transparent from the user's point of view. Figure 24 depicts the CUP step by step.

After connecting to the WO's server, the unit sends its AC and the value of $R2 = Att(CFG)$ along with a nonce C encrypted by the WO's public key. The WO's public key is obtained a priori through a secure protocol. If this is a new unit, the value of R2 is null.

Upon receiving the AC, the WO verifies if the AC is null. If the comparison is positive, the unit is a clone and the WO terminates the connection. Otherwise, the CUP continues its normal flow.

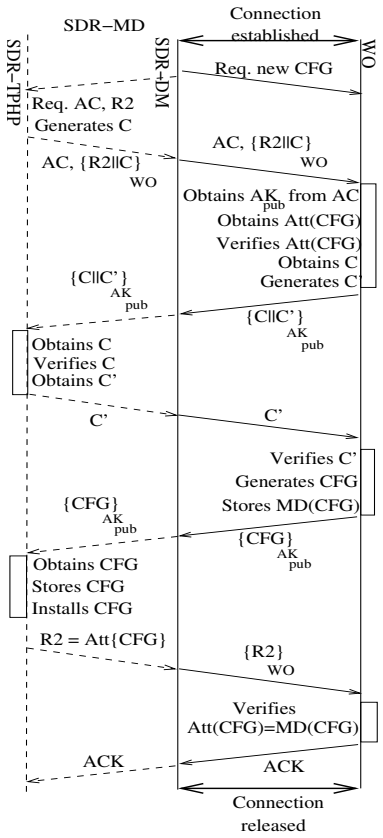

Figure 24: The CFG Update Protocol.

The WO uses the Privacy CA's public key and decrypts the AC, obtaining the AK_{pub}. The WO has a database Wireless Operator's Database (WO-DB), indexed by the AK_{pub}, which contains information about each SDR-MD in a valid state, such as phone number and user name.

Next, the WO looks for a matching AK_{pub} in the WO-DB. If it finds a match, it verifies $MD(CFG) = R2$. If the comparison is negative, this is an invalid unit; either this is a cloned unit or a masquerade attack is occurring, and counter-measures are taken.

On the other hand, if the comparison is positive, this is a valid unit. The WO then obtains C and generates a nonce C' to authenticate the unit. C is concatenated with C' and sent, encrypted by the AK_{pub}, to the SDR-MD. If the AK_{pub} is not in the WO-DB, this is a unit in the temporary state.

Upon receiving $\{C||C'\}_{AK_{pub}}$ from the WO, the unit authenticates the WO if the received C is equal to the one previously generated. If authentication fails, the SDR-MD terminates

the connection. Otherwise, it sends C' back to the WO.

Next, the WO authenticates the unit by verifying C'. If authentication fails, the WO terminates the connection. Otherwise, the WO generates a new CFG and stores the MD(CFG) value in the WO-DB. The unit receives the CFG encrypted by its AK_{pub} and decrypts it. The unit then stores the CFG in the protected storage of TRC2 and installs the new phone number.

Next, the AM Eng measures Att(CFG) and writes the value in R2. The unit then sends this value encrypted by the WO's public key to the WO. The WO verifies the value and acknowledges the unit if the comparison is positive. Otherwise, it informs the unit that an error occurred during the CFG installation step. This step is repeated in the case of errors. After receiving an acknowledgment, the unit releases the connection.

7.2.3 Valid State

After obtaining the AC from the Privacy CA and the CFG file from the WO, the SDR-MD finally reaches a valid state. Therefore, the unit is ready to use all the services offered by the WO. Figure 25 depicts the tamper-protected hardware package when the SDR-MD is in the valid state.

Note that the clone signal, sent by the AC Eng, propagates outside the TPHP to the CPU and inside the TPHP to the TRC2, where it sets the AC to null if the SDR-MD is a cloned unit.

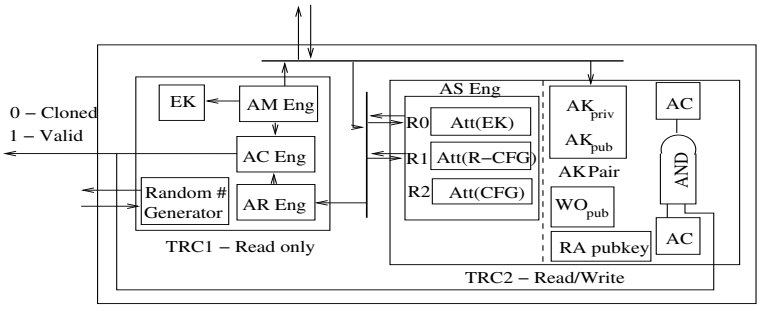

Figure 25: The tamper-protected hardware package in a valid state.

7.2.4 Cloning Aware Procedure

The cloning-aware procedure is implemented in both sides, the WO and the SDR-MD, and is responsible for detecting whether the SDR-MD is a valid unit or a cloned unit.

After the unit has connected to the WO and requested a service, the cloning-aware procedure starts in the SDR-MD side. New Att(EK) and Att(CFG) values are measured by the AM Eng and sent to the AC Eng, which also receives the current value of R0 and R2 from the AR Eng. The AC Eng compares the values and signalizes 1 for a valid unit, if $Att(EK) = R0$ and $Att(CFG) = R2$, or 0 for a cloned unit, if $Att(EK) \neq R0$ or $Att(CFG) \neq R2$. In this fashion, the SDR-MD is aware of cloning. Figure 26 illustrates the procedure. If the SDR-MD is a valid unit, the AC is sent and the WO cloning-aware procedure begins.

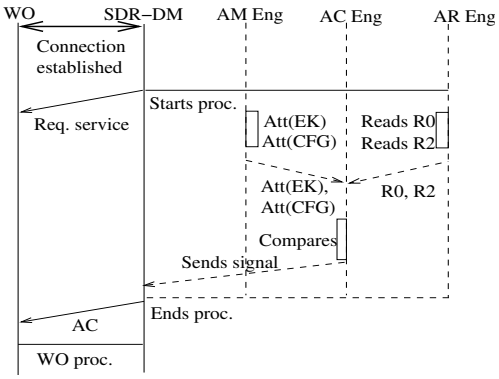

Figure 26: Cloning-aware procedure: SDR-MD side.

In the WO side, the procedure works basically as an authentication module. The WO obtains the AC and verifies if it is valid or null. If the AC is null, the WO terminates the connection, since the unit is a clone. Otherwise, the WO obtains the AK_{pub} from the AC and looks for a match in the WO-DB. If there is no match, the service is denied. If there is a match, the WO prepares to authenticate the unit.

If the unit is correctly authenticated, the WO allows the use of the service. On the other hand, if the unit is not authenticated, the WO concludes that this unit is trying to use other unit's AC (masquerade attack) and denies the service. Figure 27 illustrates the procedure.

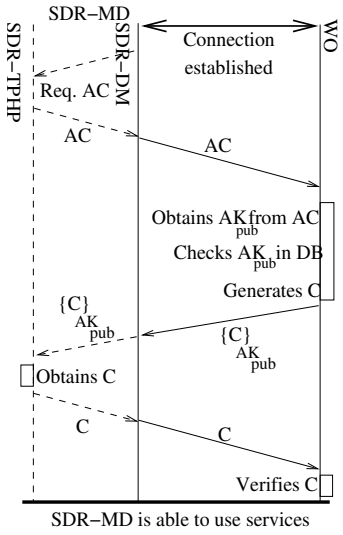

Figure 27: Cloning-aware procedure: WO side.

7.3 Formal Proofs

In this section, the correctness proofs are presented. It begins with 3 lemmas. The first lemma shows that only an SDR-MD with a valid EK is provided an AC. The second lemma shows that an SDR-MD only obtains a new CFG when its identity is successfully proved. Finally, the third lemma shows that only valid CFGs, i.e., CFGs that have been generated and signed by the WO, can be installed by an SDR-MD.

The section continues with 2 final theorems. The first theorem proves that there is no possibility to clone an SDR-MD over the air. The second theorem guarantees that only a valid SDR-MD can use the network services.

Lemma 1 *The Privacy CA only attests the identity of the SDR-MDs that have valid EKs.*

Proof: *Since the Privacy CA has a database of valid EKs and this database is assumed to be secured stored, any SDR-MD that requests an AC and sends an invalid MD(EK) value, i.e., hash of an EK that is not generated by the Manufacturer, has the AC denied.*

A replay attack is not possible since the ACP is executed only once per each EK. Impersonation of the SDR-MD, i.e., masquerade attack, is noticed by the authentication step. □

Lemma 2 *No SDR-MD obtains a CFG file unless its identity is successfully proved.*

Proof: *According to the CUP definition, only after being authenticated by the WO, the SDR-MD is given a new CFG. This eliminates the possibility of masquerade attacks and replay attacks.*

Only after responding correctly to the challenge generated by the WO, the SDR-MD is given a new CFG. Therefore, no SDR-MD obtains a new CFG file unless it has proved its identity. □

Lemma 3 *Only valid CFG files are installed in each SDR-MD.*

Proof: *To install a new CFG the SDR-MD must execute the CUP. According to the CUP definition, before receiving a new CFG the SDR-MD authenticates the WO. Thus, masquerade and replay attacks are eliminated.*

After authentication, the SDR-MD receives a new $CFG = [Phone\#]_{WO}$. *Since masquerade and replay attacks fail, only the WO could have sent this message. Thus, the CFG is considered valid and the TPHP stores and installs the new CFG.* □

Theorem 1 *It is guaranteed that there is no possibility to clone an SDR-MD over the air.*

Proof: *To clone an SDR-MD over the air, one attacker must obtain the EK of the victim or a combination of valid AK pair, valid AC and valid CFG.*

Since the EK and $AK_{private}$ *are never disclosed by the TPHP, the attacker has no possibility to obtain the EK nor the AK pair of a victim. Since the CFG is signed by the WO, it cannot be duplicated. Besides, according to Lemma 2, the attacker must prove its identity to obtain a valid CFG. Thus, if the attacker uses an AC that is not his/her, the WO will notice it and deny a new valid CFG.*

With no other way to clone an SDR-MD over the air, the only way to bill someone else's account is to capture his/her AC when transmitted over the air. However, the WO cloning-aware procedure will detect that the captured AC does not belong to that unit and it will deny any services. □

Theorem 2 *It is guaranteed that if cloned units exit, they will not be able to use the Wireless Operator services.*

Proof: *According to the WO cloning-aware procedure, in order to use the network services, the SDR-MD must present a valid AC. By Lemma 1, only SDR-MDs with valid EKs are able to obtain a valid AC. Therefore, an unit with an invalid EK does not have a valid AC and cannot use the WO's services.*

According to Theorem 1, there is no way to clone an SDR-MD over the air and impersonation of other SDR-MDs by capturing their AC is noticed by the WO cloning-aware procedure. Thus, the only other way to clone an SDR-MD is to have physical access to its TPHP.

However, if an attacker successfully disassemble the TPHP without damaging it and is able to copy the TPHP to another SDR-MD's TPHP, Lemma 3 and the SDR-MD cloning-aware procedure guarantee that the SDR-MD which received the cloned TPHP denies the use of the network services. The value of R2 on the cloned TPHP and the value of the current MD(CFG) in the device are different. Thus, the SDR-MD blocks the use of any services.

Since the SDR-MD cloning-aware procedure blocks the use of any services by cloned units and the WO cloning-aware procedure notices masquerade attacks, it is guaranteed that only a valid SDR-MD can use the Wireless Operator services. □

In summary, the fraud-prevention framework elevates the level of difficulty to clone an SDR-MD. The only way to clone one SDR-MD that employs the framework would be disassembling the TPHP from the SDR-MD and reading its contents. Since the TPHP is physically bound to other parts of the SDR-MD, attempts to disassemble it would probably damage the TPHP. Even if an attacker successfully disassembles the TPHP without damaging it, the equipment to read and copy the TPHP is so expensive that the attacker would practically have no gain, if any, in doing so.

7.4 Experiments

This section presents comparison experiments and remarks regarding the anti-cloning scheme and the GSM authentication scheme. The setup used is depicted in Figure 28. The entities

involved in this experiment are specified in the list below:

1. The Privacy CA and WO servers: Toshiba Pentium 4 with CPU speed of 3.06 GHz, 1GB RAM, and Linux OS.

2. The client: a Sharp Zaurus PDA CL-760 with CPU speed of 400 MHz, 64MB SDRAM, Linux OS, and J2ME support.

3. The router: a Netgear 108Mbps wireless firewall router. Model WGT624.

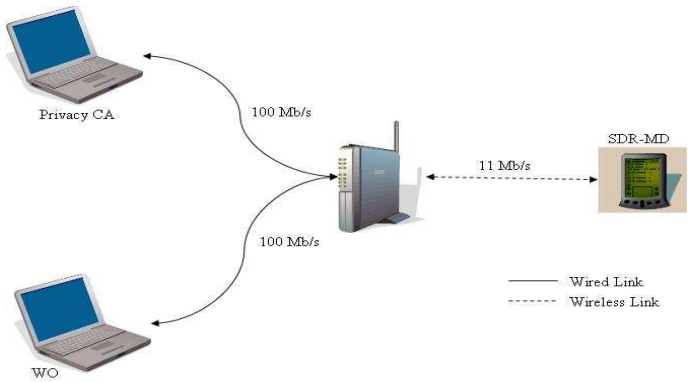

Figure 28: The experiments setup

In the first experiment, the total time for an SDR-MD to reach the valid state is calculated. Then, in the second experiment, a comparison between the CUP and the current method of changing a cellular phone number is presented. Finally, in the third experiment, the time to execute the cloning aware procedures is calculated and a comparison with the GSM authentication scheme is performed.

In all experiments the SHA-1 algorithm is employed as the hash functions, the EK is an AES 128-bit key, and the public/private keys used are RSA 128-bit keys, in a first moment, and RSA 512-bit keys, in a second execution test. The public/private keys vary their sizes so that a comparison when using different key sizes can be performed.

7.4.1 Experiment 1 - Reaching the Valid State

In this experiment, the total time for a recently purchased SDR-MD to reach the valid state is calculated. A comparison when using different RSA keys for the public/private keys employed is also made.

The graph in Figure 29 shows the performance of the ACP, the CUP, and the total time to reach the valid state when using RSA 128-bit and 512-bit keys. Note that the time difference when using 128-bit keys and 512-bit keys is minimal.

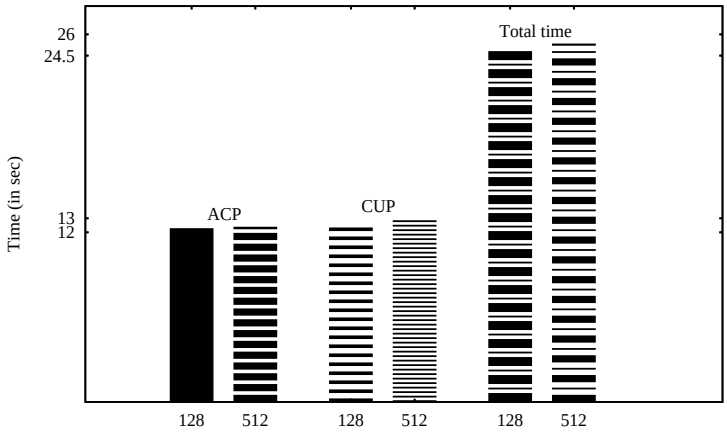

Figure 29: Time to reach the valid state.

Remember that the ACP is executed only once per each EK and before the user is able to use the SDR-MD. Therefore, the performance of this protocol is not as important as the CUP, as long as it does not take too long.

As it can be seen, the time to complete the ACP, when using 128-bit public/private keys, is around 12 seconds. While when using 512-bit public/private keys, it takes around 13 seconds. This difference is minimal, and depending on the cloning aware procedures performance, it is indicated to employ 512-bit keys.

The same can be observed about the CUP's performance. There is again practically no difference when using 128-bit keys or 512-bit keys.

An important remark in both protocols is the time to generate the 48-bit random number. The device used in this experiment does not have the specialized random number generator hardware, thus the time to generate the random number is by far the step that takes longer, around 11 seconds. That represents more than 90% of the total time to perform the ACP and the CUP when using 128-bit keys, and over 80% when using 512-bit keys.

7.4.2 Experiment 2 - Time to Change the Device's Phone Number

This experiment presents a comparison between the total time to change the device's phone number when using the CUP, presented in this scheme, and the current method that is used by Wireless Operators.

The first point to be addressed is the transparency to the final user. Remember that the CUP is totally transparent, there is no need for the user to iterate with the protocol. The CUP will provide, download, install, and update both the register in the SDR-MD and the WO database. The only role the user plays is to initiate the protocol by sending a request to the WO.

On the other hand, the final user plays a major role in the current method employed to change cellular phones. The user has to call the WO, obtain a new phone number, program him/herself this new number in the phone and wait for several minutes for the WO to update its database.

Let's assume for a minute that the user is actually familiar with this process, and that the WO attendant is experienced. After calling the WO, the user is put in a wait line. Then, the user finally talks to someone and explains what he/she wants. After a couple more minutes the new phone number is obtained, and the user has to program him/herself the phone number in the device. When the process seems to be over, the user still has to wait for 30 minutes to one hour to use the device. This is the time necessary to have the WO's database updated.

The second point to be addressed is the superior performance of the CUP when comparing with the current method to update the device's phone number. The results are presented in Table 9. Note that it takes less than 15 seconds to update the device's phone number when using the CUP and more than one hour when using the current method.

Table 9: Time to update the device's phone number.

CUP 128-bit keys	12.5 sec
CUP 512-bit keys	13 sec
Current Method	15 min call + 30 to 60 min database update

7.4.3 Experiment 3 - Authentication Time

To complete the authentication step, the cloning aware procedures must be executed in both sides the SDR-MD and the WO. First, the SDR-MD executes its cloning aware procedure to discover if it is a valid or cloned unit, it basically authenticates itself. Then, the WO executes its cloning aware procedure to authenticate the SDR-MD and avoid the use of the services by cloned units.

Since the cloning aware procedures are invoked every time the SDR-MD tries to use the network services, performance in this part is by far more important than in any other part of the anti-cloning scheme.

Remember that the SDR-MD cloning aware procedure is basically a comparison of the current values stored in R0 and R2, and the current values of the functions Att(EK) and Att(CFG). On the other hand, the WO cloning aware procedure works as an authentication module and requires decryption operations on the SDR-MD side.

The graph in Figure 30 depicts the results when using 128-bit and 512-bit public/private keys. Note that of the total time to authenticate, about 60% is spent on the WO procedure. This is because the network is actually being already used, i.e., authentication data is being transmitted, and the SDR-MD has to perform decryption operations.

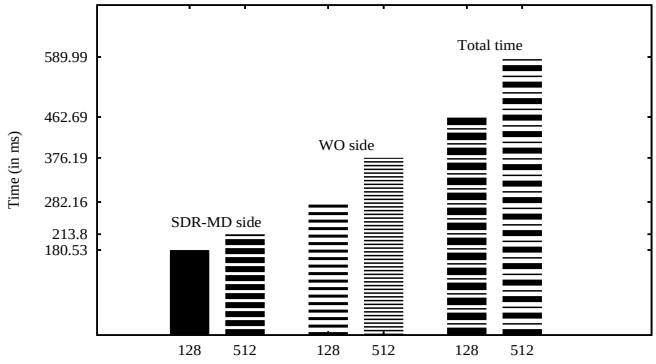

Figure 30: Authentication time.

Although the time to perform authentication in the GSM scheme is not explicitly mentioned, the total time to obtain the carrier tone when making a call is around 4 seconds in average [29].

Suppose that 4 seconds would also be the time to obtain services when using SDR-MDs. Then, the authentication time when using 512-bit keys is around only 15% of the total time to obtain the requested services. Therefore, the cloning aware procedures achieved satisfactory performance and the use of 512-bit keys is indicated.

CHAPTER VIII

A SECURE AND EFFICIENT PROTOCOL FOR RADIO CONFIGURATION FILES DOWNLOAD AND UPDATE

As mentioned in previous chapters, SDR-MDs are constrained devices. They do not have much processor power, memory, storage space, and long battery life. However, to be capable of reconfiguring their radio parameters and switching modes, they need to download, store, and install different R-CFGs.

If this process is too long, the SDR-MD might have its service disconnected during a handover. However, if the SDR-MD installs invalid R-CFGs or malicious data, skipping the security checks, it might start presenting malfunction, functioning outside its defined spectrum, interfering in other devices' communication or simply losing important data.

The proposed secure and efficient protocol employs some of the ideas presented in previous chapters to establish the minimal necessary steps to download and update the current R-CFG in a safe and efficient manner.

Security safeguards, presented in Chapter 5, are implemented to prevent attacks that seek to install malicious code. The light differential download algorithm, presented in Chapter 6, is employed to compress the R-CFG, saving time in the downloading step, and minimizing transmission errors.

The LSSL protocol, presented in Chapter 3, is an option to protect the manufacturer's server and SDR-MD connection, and avoid eavesdropping of proprietary information. Another option is to encrypt the R-CFG with the SDR-MD's attestation public key and use the AC to authenticate the SDR-MD, as performed by the anti-cloning scheme.

Experiments comparing the proposed protocol when using LSSL, SSL, and the AC+AK_{Pub} combination are presented. The protocol can also be found in [6].

8.1 The Protocol Specification

The proposed protocol specifies the steps and messages transmitted during the communication of the SDR-MD with the manufacturer's server when downloading a new delta set. For simplicity the protocol is divided into 5 modules. Figure 31 depicts the proposed protocol when no errors occur.

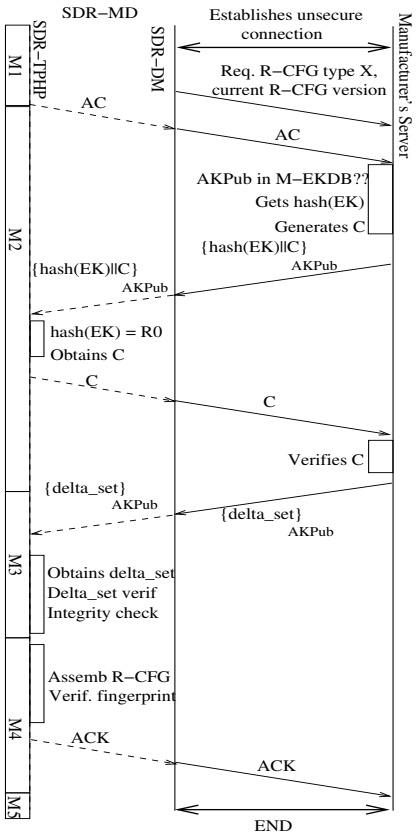

Figure 31: Secure and efficiente protocol for SDR download and update.

The first module (M1) establishes a connection that can be either unsecured (using sockets) or secured (LSSL or SSL). The protocol is specified using the AC+AK_{Pub} combination. The use of the LSSL or the SSL is obtained by similarity. Still in this module, the SDR-MD also presents its current R-CFG version and requests the most up-to-date R-CFG.

During the second module (M2), the steps of mutual authentication are performed. The SDR-MD presents its AC to the manufacturer's server, which will then obtain AK_{Pub}, find hash(EK) in its database, generate C (random number), and send $hash(EK)||C$ encrypted by the AK_{Pub}.

The SDR-MD obtains the message, checks if $hash(EK) = R0$, authenticating the Manufacturer's server if the outcome is positive and sends C back. The manufacturer verifies C and authenticates the SDR-MD. If the LSSL or the SSL is employed, instead of the AC+AK$_{Pub}$ combination, then the mutual authentication step is done by those protocols.

In the third module (M3), the SDR-MD obtains the delta set and performs the security computations that verify and check data integrity of that file.

In Module (M4), the new R-CFG is assembled and has its integrity checked. The original data integrity value is obtained in the delta set header. The SDR-MD calculates the new R-CFG fingerprint and compares with the original value. If the outcome is positive, then the SDR-MD sends an acknowledgement to the server and prepares to install the R-CFG. On the other hand, if the comparison result is negative, the SDR-MD sends an error message to the server.

Module 5 (M5) gracefully releases the connection. There is a reconfiguration/installation module but this step occurs after the download process is completed and falls outside the scope of this work.

8.2 Protocol Structure Analysis

The protocol defines 5 distinct messages:

VER - for the SDR-MD's current R-CGF version

REQ_Y - to request a new R-CFG type Y

ACK - for messages combined with a positive acknowledgment

ERR_X - for messages with transmission errors. X is the index of the error that varies from 1 to 13;

DATA - for data packets (delta set);

END - to release connection.

Next, each module is individually analyzed according to the messages defined. Timeout periods are not shown during the analysis, but in case a timeout period has expired, the

previous message is resent.

Module 1: The connection is established, the SDR-MD informs its current R-CFG version, and requests a new R-CFG. The following errors can occur during M1:

1. ERR_1 - Connection could not be established.
2. ERR_2 - Current R-CFG version does not exist. Sent from the server to the client.
3. ERR_3 - Current R-CFG version is the most up-to-date. Sent from the server to the client.

Module 2: Mutual authentication takes place. The following errors can occur during this module:

1. ERR_4 - AK_{Pub} not in M-EKDB database. Sent from the server to the client.
2. ERR_5 - Internal error on the server. It was not possible to encrypt the authentication message. Sent from the server to the client.
3. ERR_6 - Internal error on the SDR-MD. It was not possible to decrypt the authentication message. Sent from the client to the server.
4. ERR_7 - Server's authentication failed. Hash(EK) $\neq$ R0. Sent from the client to the server.
5. ERR_8 - Client's authentication failed. Received C $\neq$ original C. Sent from the server to the client.

Module 3: The SDR-MD receives the delta set and performs the security checks. The following errors can occur during this module:

1. ERR_9 - Internal error on the server. It was not possible to encrypt the delta set. Sent from the server to the client.

2. ERR_10 - Internal error on the SDR-MD. It was not possible to decrypt the delta set. Sent from the client to the server.

3. ERR_11 - Delta set verification or data integrity check has failed. Sent from the client to the server.

Module 4: The SDR-MD assemblies the R-CFG and checks its fingerprint. The following errors can occur during this module:

1. ERR_12 - Internal error on the client. It was not possible to assemble the R-CFG. Sent from the client to the server.

2. ERR_13 - Fingerprint comparison has failed. Original fingerprint $\neq$ calculated fingerprint. Sent from the client to the server.

Module 5: In M5 the connection is finally released. If the protocol has reached this module, it has properly terminated.

Table 10 illustrates some common methods of attacks that fail against the proposed protocol.

8.3 Consistency

A protocol is said to be consistent if there are no deadlocks, livelocks and terminates properly [19]. To prove that the secure and efficient protocol is deadlock and livelock free a theorem is formulated.

The protocol's state machine is illustrated in Figure 32. Each state represents one module in the protocol and an arrow from one state to another indicates that the protocol's execution can successfully flow from this module to the next one with a certain transition probability.

The probability to go from the M1 to M2 is P_{12}, the probability to loop in M2 is P_{22}, and so on. The probability of correct terminating the protocol in M5 is P_t. Under normal conditions, the probability that the protocol's normal execution flow will occur is higher

Table 10: Possible attacks and how the protocol secures them.

Attacks	**Description**	**Protection**
Access control	Clients using unauthorized services or trying to download data they should not.	Protocol employs client authentication.
Masquerade	An entity pretends to be the Manufacturer server or a client.	Protocol uses mutual authentication.
Confidentiality	R-CFG might be confidential.	By establishing secure connections or encryption/decryption mechanisms, proprietary information is kept secret.
Replay	Messages are captured and retransmitted later.	Mutual authentication will prevent replay attacks. Furthermore, the protocol uses timestamps during the LSSL.
R-CFG Verification	Installing R-CFGs that are not approved by the RA.	Every delta set is digitally signed by the RA.
R-CFG Integrity	R-CFG modified after it has been approved.	Protocol employs one-way hash functions to guarantee data integrity.

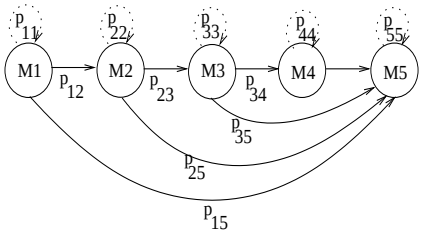

Figure 32: Protocol's state machine.

than any other flows. Table 11 shows the probabilities to go from one module to another under normal condition.

The dotted arrows represent an internal loop in each module due to a timeout repetition. Suppose, for instance, that in M2, a timeout expires after the R-CFG version is informed. The SDR-MD re-sends the VER message and waits for a response. This can lead to an infinite loop if the timeout happens indefinitely.

To avoid infinite loops a timeout counter is included in the protocol. Each time a timeout occurs, the counter is incremented. Each time a message is received, the counter is reset to 0. If the counter reaches a certain number X (i.e., the timeout has occurred X times consecutively), it probably means the network is too congested. In this case, the device

Table 11: Probabilities to go from one module to another.

$P_{12} > P_{11}$	$P_{12} > P_{15}$
$P_{23} > P_{22}$	$P_{23} > P_{25}$
$P_{34} > P_{33}$	$P_{34} > P_{35}$
$P_{45} > P_{44}$	
$P_t > P_{55}$	

simply terminates the connection without going to M5.

Note that in Figure 32 there are no infinite loops or indefinite waiting due to physical resources. In fact, the only resource in the protocol is the communication channel. Therefore, the protocol is trivially deadlock and livelock free.

Whenever the protocol execution reaches M5, the protocol is properly terminated. Improper terminations may occur in three different stages: before, during, or after the delta set download. If it occurs before or during the download, whatever caused the download to occur the previous time will most likely remain and therefore cause the download to occur again. If it occurs after the download is completed, the manufacturer will be informed whether the device has accepted the new R-CFG when it connects to the manufacturer's server again. Otherwise, M5 terminates the session gracefully.

Theorem 3 verifies the proper termination property of the proposed protocol.

Theorem 3 *The protocol properly terminates if each message is transmitted with a bounded delay d, which is less than the timeout value used in the protocol.*

Consider a message, m_i, being transmitted by the device at time t_1. Let t_2 be the time that the server correctly receives m_i and t_3 the time at which the device sends m_{i+1}. Showing that $t_1 < t_2 < t_3$ and that t_3 is less than the timeout value, is sufficient to demonstrate liveness, since by induction each message will be transmitted with a bounded delay d.

Let $R(t)$ be the received sequence number as a function of time (at the server) and $S(t)$ be the transmitted sequence number (at the SDR-MD). Now, let $N(t)$ be the sequence number of the next expected transmission ($N(t) = R(t)+1$). Since $S(t)$ is the largest request number received from the server up to time t so $S(t) \leq N(t)$ and $N(t) \leq i$.

Because $R(t)$ is incremented to $i + 1$ at t_2, $S(t)$ is incremented to $i + 1$ at t_3 and $S(t) \leq N(t)$, it follows that $t_2 < t_3$. The device transmits m_i repeatedly, with a finite

timeout between retransmission, from t_1 until it is first received error-free at t_2. Since there is a probability $p > 0$ that each transmission is received correctly, an error-free reception eventually occurs and t_2 is bounded and less than the timeout. t_3 can be obtained using a similar argument from the server side. □

The server side consistency can be obtained by the application of similar ideas discussed above.

8.4 Experiments

In this section five practical experiments are presented. The setup of these experiments is the same as in Figure 4. The Sharp Zaurus PDA CL-760 connects to the Dell Pentium 4 - 2.6GHz through the Netgear 108Mbps wireless firewall router.

In the first experiment, the time to execute each module individually is calculated. The second experiment compares the time spent to perform the security checks in the delta set against the time to check the entire R-CFG.

In the third experiment, the protocol is executed employing the LSSL, then the SSL, and finally the AC+AK_{Pub} combination. The time it takes to complete the secure and efficient protocol in each execution is compared.

The fourth experiment presents a quick analysis of the differential download step. The goal is to find in which scenarios the protocol should employ the LDDA and in which scenarios the protocol should transfer the entire R-CFG.

Finally, the final experiment simulates a handover process. The SDR-MD is handed over from a 802.11b network to another 802.11b network. The time to execute the secure and efficient protocol plus the time to install the R-CFG are computed.

8.4.1 Experiment 1 - Module by Module

In this experiment, the SDR-MD connects to the manufacturer's server employing the AC+AK_{Pub} combination, just like Figure 31. The time to perform each module of the secure and efficient protocol is measured.

Table 12 presents the results when using 128-bit keys for the public/private keys, SHA-1

to calculate the hash value of an element, and transferring different delta sets.

Table 12: Execution time of each module.

	1M+128	1M+256	1M+512	1M+1M
M1	66.15 ms	66.15 ms	66.15 ms	66.15 ms
M2	156.7 ms	156.7 ms	156.7 ms	156.7 ms
M3	2.22 sec	3.81 sec	7 sec	13.5 sec
M4	5.85 sec	6.2 sec	7.37 sec	9.31 sec
M5	10 ms	10 ms	10 ms	10 ms
Total	8.29 sec	10.24 sec	14.6 sec	23 sec

Note that, as expected, M1, M2, and M5 have constant execution time. They are independent of the delta set size. On the other hand, M3 and M4 perform operations on the delta set, so their execution time varies according to the delta set size.

8.4.2 Experiment 2 - Comparing the Security Checks Time

The goal in this experiment is to show how much time is saved by performing the security checks in the delta set instead of the entire R-CFG as proposed in Chapter 5.

A 1MB R-CFG is used as the old R-CFG. This old R-CFG is subjected to be upgraded employing a delta set of 128KB, 256KB, 512KB and finally a 1MB delta set.

The graph in Figure 33 depicts the results. Notice that it takes more than the double of the time to perform the security checks in the entire R-CFG than the time spent to do it in the delta sets. Moreover, the older the R-CFG, the bigger the difference in performance.

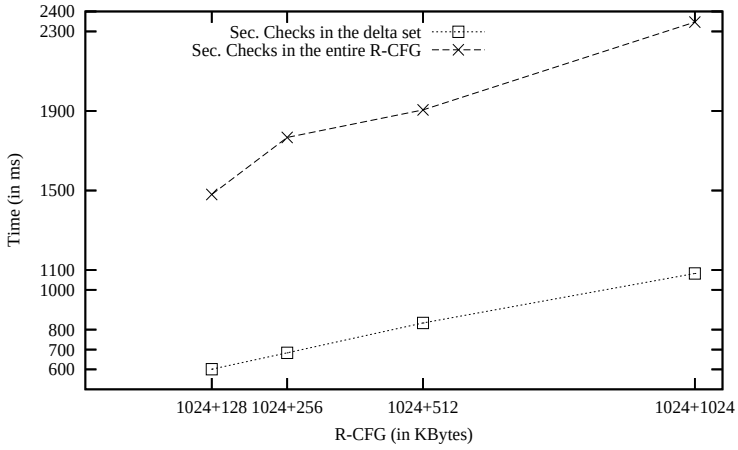

Figure 33: Comparing the security checks performance.

8.4.3 Experiment 3 - Comparing the Protocol's Execution Time

In this experiment, the total time to execute the secure and efficient protocol when employing the LSSL (Prot + LSSL), the SSL (Prot + SSL) or the AC+AK$_{Pub}$ combination (Prot + AC + AKPub) is compared. The time to transfer the delta set file (Socket no encryption) and the time to transfer the entire raw R-CFG (Raw R-CFG no enc) over an unsecured connection are also presented.

The graph in Figure 34 shows the results. Note that as expected the best performance regarding completion time is obtained when transferring the delta set file over an unsecured connection.

This method is about 20% to 40% faster than when using the LSSL or the AC+AK$_{Pub}$ combination, 40% to 50% faster than when using the SSL, and 60 % to 70% faster than when transmitting the raw R-CFG. This is because there is no need to spend time with encryption and decryption. This method is indicated if there is no proprietary information in the R-CFG files.

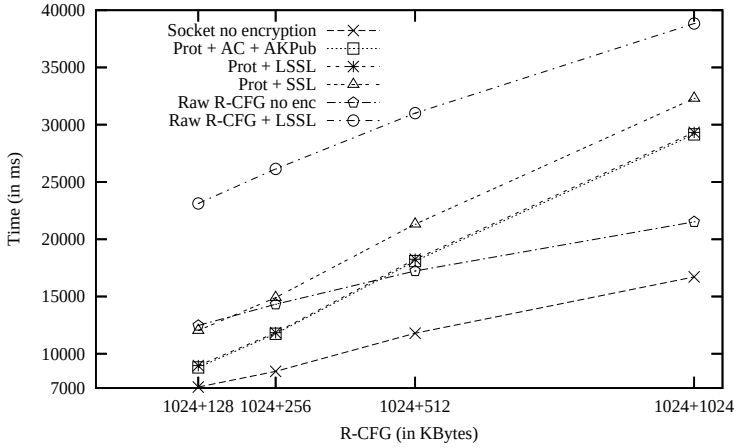

Figure 34: Comparing the protocol with different techniques.

The performance when using the AC+AK$_{Pub}$ combination and the LSSL is basically the same. This is expected since they perform similar operations. The SSL performs a little worse, about 10% to 25% slower, since it is not specified for constrained devices.

Finally, the worst performance is obtained when a method that does not employ the LDDA is executed. It takes longer to transfer the entire R-CFG than transferring the delta

set, performing encryption/decryption operations, and assembling the final R-CFG in the SDR-MD. This shows again that compressing the R-CFG is of great importance.

The results in experiments 1 and 2 show that the performance to complete the secure and efficient protocol is satisfactory when the SDR-MD is changing modes but not when using the network services. For instance, it takes less than 20 seconds to securely download and assembly the new R-CFG, when using the LSSL or the AC+AK_{Pub} combination, in the 1M+512 case.

However, the protocol's execution time is not enough to ensure that the SDR-MD will not have its services disconnected during the handover procedure. Specially if the current R-CFG version is too old. Therefore, methods to improve even more the performance of the secure and efficient protocol are necessary.

8.4.4 Experiment 4 - Differential Download Performance Analysis

This experiment presents a quick analysis of the differential download step of the secure and efficient protocol. The goal is to find when the differential download step should or should not be performed. Therefore, this experiment searches for the borderline of when the protocol should employ the LDDA and when the protocol should transfer the raw new R-CFG.

To obtain the borderline rate, the old and new R-CFG sizes are fixed in 1MB and the data file size varies from 12.5% of the new R-CFG size to 100%. The time to transfer the raw new R-CFG and perform the security checks is plotted against the time to transfer the delta set, perform the security checks, generate the new R-CFG and verify its integrity.

The experiment compares the performance of a method employing the secure and efficient protocol with the LSSL (Prot + LSSL), a method employing the secure and efficient protocol but not protecting the communication channel (Socket no enc), a method transferring the raw new R-CFG but not protecting the communication channel (Raw R-CFG no enc), and finally, a method transferring the raw new R-CFG employing the LSSL to protect the communication channel (Raw R-CFG + LSSL).

The graph in Figure 35 shows the comparison results. As it can be seen, when there

is no need to protect the communication channel, i.e., when the R-CFG does not contain proprietary information, if the data file is 55% or more of the new R-CFG size, it is faster to transfer the raw new R-CFG (Raw R-CFG no enc) than using the LDDA (Socket no enc). In cases that there is a need to protect the communication channel, the borderline is about 80%. Therefore, the protocol should employ the LDDA (Prot + LSSL) whenever the data file size is 80% or less than the new R-CFG size (Raw R-CFG + LSSL).

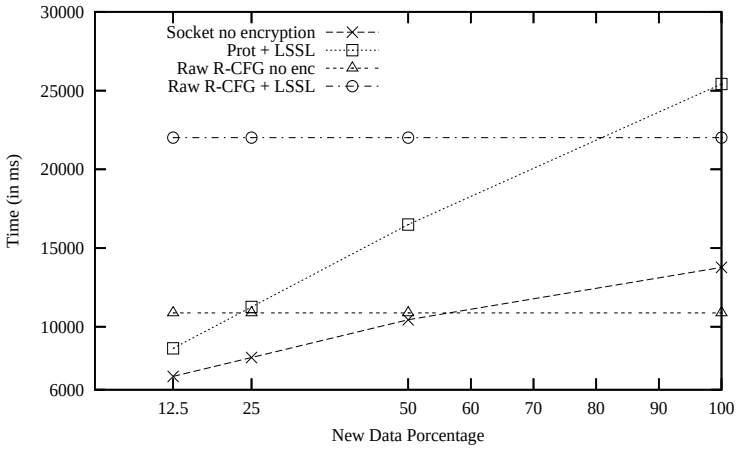

Figure 35: Finding the borderline rate to employ the LDDA.

8.4.5 Experiment 5 - A Handover Simulation

In this experiment, a handover is simulated. The SDR-MD is handed over from a 802.11b network, called network A, to another 802.11b network, called network B. The time to perform the secure and efficient protocol plus the time to install the new R-CFG are computed.

To simulate the handover, the Linux *iwconfig* command is employed. This command makes it possible to reconfigure the network card parameters without the need to restart the device. In this experiment, the iwconfig is used to reconfigure the network name (ESSID) and the frequency/channel.

At first, the SDR-MD is connected to network A. It requests and downloads the delta set, performs the security checks, generates the new R-CFG, and installs the new R-CFG.

After completing the secure and efficient protocol, the SDR-MD finally executes the new R-CFG and has its network name and frequency/channel changed. Since the experiment is performed in a more high level, i.e., the new R-CFG is not loaded into the FPGA and the

reconfiguration only occurs in the network card, all the FPGA commands are ignored by the SDR-MD's device manager. Only the iwconfig command is executed.

The graph in Figure 36 shows the results for this experiment. The delta set size is around 128KB and the old and new R-CFG sizes are equal to 1MB. Note that the execution of the secure and efficient protocol is around 71% of the total time, while the reconfiguration step is around 39% of the total time.

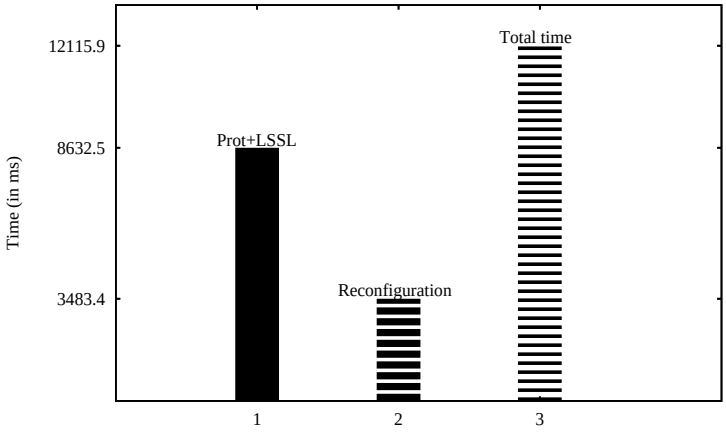

Figure 36: A handover simulation.

CHAPTER IX

CONCLUSION

The software defined radio technology has been successfully introducing the concepts of programmability and reconfigurability to improve telecommunication networks.

Systems with higher flexibility and openness have been proposed making the mobile communication environment much more dynamic. Clearly, these new concepts have great advantages since the customer is able to use the same device in different parts of the world, with different wireless technologies. Thus, being able to maximize the quality of service and minimize cost.

Despite the advantages, there are still issues to be discussed regarding security. Some of the concerns are the radio configuration download, storage and installation, the user's privacy, and SDR-MD cloning.

To address those concerns and greatly enhance the overall security of SDR-MDs and their new highly dynamic environment a fraud-prevention framework was presented.

The framework offers security monitoring against malicious attacks and viruses that may affect the configuration data; protects sensitive information through the use of protected storage; creates and protects an identity for the system; employs a secure and efficient protocol for radio configuration download and update; and finally, establishes an anti-cloning scheme, which not only guarantees that no units can be cloned over the air, but also elevates the level of difficulty to clone units if the attacker has physical access to the SDR-MD. Even if cloned units exist, the anti-cloning scheme is able to identify and deny services to those units.

The research contributions of the framework can be summarized, in crescent order of importance, as follows:

- A light secure socket layer protocol

- Security measures to download and install radio configuration files
- A light differential download algorithm
- A secure and efficient protocol to download and update radio configuration files
- An anti-cloning scheme

9.1 A Light Secure Socket Layer Protocol

The LSSL protocol was presented as a mean to obtain secure connections between servers and SDR-MDs. The protocol is based on the SSL; however, it is more suitable for SDR-MDs operating under low-capabilities, low-bandwidth, and error-prone wireless links, since the computational intensive operations are performed in the server side.

Although the LSSL provides less cipher suites than the SSL, experiments showed that it works with the most common encryption/decryption algorithms and it is unlike that the LSSL would not work for a combination of servers/SDR-MDs that the SSL would.

The LSSL's main idea of performing intensive operations in the server side confirmed to be effective. The LSSL showed to be about 2.1 seconds faster than the SSL for all cases tested. The difference in performance is constant because the savings are obtained during the initial setup, which is independent of the amount of data to be transmitted.

9.2 Securing the Download and Installation of R-CFGs

Security measures were presented to download and install R-CFGs in a safe manner. Public/private keys and hash functions were employed to validate and verify the R-CFGs. The validation process is performed by the regulatory agency, which tests, approves, and signs the R-CFG if no malfunction is detected.

The SDR-MD downloads the R-CFG and performs the R-CFG verification and data integrity check. Basically, the SDR-MD checks the RA signature and if the file has not been tampered with after being approved by the RA.

Experiments showed that the larger the R-CFG is, the longer the SDR-MD takes to perform the security checks. It takes, for instance, about 600 milliseconds to verify a

128KB R-CFG, about 1.08 seconds to verify a 1MB R-CFG, and about 2.35 seconds to verify a 2MB R-CFG. These results indicated that a compression method should be used to minimize this time and improve performance.

9.3 A Light Differential Download Algorithm

The light differential download algorithm was presented as a mean to compress the R-CFG, minimize transmission errors, and improve overall performance. The LDDA generates a delta file provided an old R-CFG and a new R-CFG. With this application, an SDR-MD downloads and performs the security check in a much smaller file.

The LDDA, which is the first differential download algorithm specifically designed for SDR download, presents several new features, such as optimization tailored for R-CFG downloads, efficient instruction logic, elimination of redundancy logic, simpler and smaller delta-sets, and independence of OS platform.

Experiments were performed comparing the LDDA and the Xdelta, another well-known differential download algorithm. The results showed that the LDDA performs better than the Xdelta, even in an unconstrained environment. A 50% improvement is achieved by the LDDA when assembling the instructions to generate the new R-CFG. A 10% to 30% improvement is achieved by the LDDA when completing the whole process (download + assembly time).

Finally, the LDDA is evaluated in a constrained environment and compared with a method that transfers the raw R-CFG. The results show that the LDDA presents a 25% to 50% improvement in performance, and that the more up-to-date the current R-CFG installed in the SDR-MD is, the better the improvement in performance is.

9.4 A Secure and Efficient Protocol to Download and Update Radio Configuration Files

A secure and efficient protocol was presented to establish the minimal necessary steps to download and update R-CFGs in a safe and efficient manner.

The R-CFG validation and verification steps were implemented to prevent attacks that seek to install malicious code. The light differential download algorithm was employed to

compress the R-CFG, saving time in the downloading step and in the security checks, and minimizing transmission errors.

In case that the R-CFGs contain proprietary information, the LSSL protocol is an option to protect the communication channel. Another option is to encrypt the R-CFG with the SDR-MD's attestation public key and use the AC to authenticate the SDR-MD.

A comparison of the total time to execute the secure and efficient protocol when employing the LSSL, the SSL, or the AC+AK_{Pub} combination was performed. The time to transfer the delta set file and the time to transfer the entire raw R-CFG over an unsecured connection were also calculated.

The results showed that the best performance was obtained when transferring the delta set file over an unsecured connection. This is because there is no need to spend time with encryption and decryption. This method is about 20% to 40% faster than when using the LSSL or the AC+AK_{Pub} combination, 40% to 50% faster than when using the SSL, and 60% to 70% faster than when transmitting the raw R-CFG. Transferring the delta set over an unsecured connection is indicated if there is no proprietary information in the R-CFG files.

The performance when using the AC+AK_{Pub} combination and the LSSL was basically the same. This was expected since they perform similar operations. The SSL performed a little worse, about 10% to 25% slower, since it is not specified for constrained devices.

Finally, the worst performance was obtained when a method that does not employ the LDDA was executed. It took longer to transfer the entire R-CFG than to transfer the delta set, perform encryption/decryption operations, and assemble the final R-CFG in the SDR-MD. This showed again that compressing the R-CFG is of great importance.

These results showed that the performance to complete the secure and efficient protocol is satisfactory when the SDR-MD is changing modes but not using the network services. It took, for instance, less than 20 seconds to securely download and assembly the new R-CFG, when using the LSSL or the AC+AK_{Pub} combination, in the 1M+512 case.

However, the protocol's execution time is not enough to ensure that the SDR-MD will not have its services disconnected during the handover procedure. Specially if the current

R-CFG version is too old.

A quick analysis of the differential download step of the protocol was also presented. The goal was to search for the borderline of when the protocol should employ the LDDA and when the protocol should transfer the raw new R-CFG. The results showed that if there is no need to protect the communication channel, whenever the data file is 55% or more of the new R-CFG size, it is faster to transfer the raw new R-CFG than using the LDDA. In cases that there is a need to protect the communication channel, the borderline is about 80%. Therefore, the protocol should employ the LDDA whenever the data file size is 80% or less than the new R-CFG size.

The final experiment simulated a handover. The SDR-MD is handed over from a 802.11b network to another 802.11b network. The time to perform the secure and efficient protocol plus the time to install the new R-CFG were computed. The results showed that for a delta set of 128KB the execution of the secure and efficient protocol is around 71% of the total time, while the reconfiguration step is around 39% of the total time.

9.5 The Anti-cloning Scheme

The anti-cloning scheme was designed to provide a core set of hardware and software technologies that establish the basis for a wireless network environment free of cloned units.

Unlike other schemes, the SDR-MD is aware of cloning. As a security measure backup, the wireless operator is also responsible for detecting if a cloned SDR-MD is trying to use the network services.

The anti-cloning scheme is composed of a tamper protected hardware package, protocols to obtain the attestation credential and the CFG file, and cloning aware procedures in both sides the SDR-MD and the WO.

Practical experiments showed that the difference of the scheme performance when using 128-bit or 512-bit public/private keys is minimal. Thus, to improve security 512-bit public/private keys should be employed.

When comparing the manner the scheme updates the SDR-MD phone number and the

current method employed by the wireless operators, the scheme showed a much more superior performance, besides being totally transparent to the final user. It took, for instance, less than 15 seconds to update the phone number when using the anti-cloning scheme and about 45 to 60 minutes with the current method.

The experiment results also showed that the total time to execute the cloning aware procedures and authenticate the SDR-MD to the WO is satisfactory. It took 463 milliseconds when using 128-bit public/privates keys and 590 milliseconds when using 512-bit public/private keys.

When compared with the total time to obtain the carrier tone in the GSM mode, when placing a call, the cloning aware procedure time would be around only 15% of the total time to obtain the requested services.

9.6 Future Work

After presenting all the practical experiment results and analyzing performance, the concerns now are to avoid disconnection of services during the R-CFG installation and the handover procedure.

The secure and efficient protocol's execution time is not enough to ensure that the SDR-MD will not have its services disconnected during the handover procedure. Therefore, there is a need to further improve the performance on this protocol without compromising security.

Methods to decrease R-CFG transmission time and R-CFG assemblage time should be the focus. The development of a light binary compression algorithm that will improve compression and save time during the transmission might be the starting point.

REFERENCES

[1] Apache organization website. *Available HTTP: http://www.apache.org*, last viewed on June 2005.

[2] Bing, B. and Jayant, N., "A cellphone for all standards," *IEEE Spectrum*, vol. 39, pp. 34–39, May 2002.

[3] Brawerman, A., Blough, D., and Bing, B., "Securing the download of radio configuration files for software defined radio devices," in *Proc. of the ACM International Workshop on Mobility Management and Wireless Access*, Oct. 2004.

[4] Brawerman, A., Blough, D., and Bing, B., "A light differential download algorithm for software defined radio devices," in *IEEE Consumer Communications and Networking Conference*, Jan. 2005.

[5] Brawerman, A. and Copeland, J., "An anti-cloning framework for software defined radio devices," in *IEEE International Conference on Communications*, May 2005.

[6] Brawerman, A. and Copeland, J., "A fraud-prevention framework for software defined radio devices.," *To be submitted to the IEEE Wireless Communication Magazine*, Aug. 2005. *Available HTTP: http://www.ece.gatech.edu/ ale/TR/TR002.pdf*

[7] Brawerman, A. and Copeland, J., "Towards a fraud-prevention framework for software defined radio devices.," *EURASIP Journal on Wireless Communications and Networking. Special Issue: Reconfigurable Radio for Future Generation Wireless Systems.*, Oct. 2005.

[8] Castelluccia, C. and Dupont, F., "Simple privacy extension for mobile ipv6." Internet Draft: Draft-Castellucia-MobileIP-Privacy, The Internet Engineering Task Force, Feb. 2001.

[9] Cleary, J. and Witten, I., "Data compression using adaptive coding and partial string matching," *IEEE Transactions on Communications*, vol. 32, pp. 396–402, Apr. 1984.

[10] Dillinger, M., Madani, K., and Alonistioti., N., *Software Defined Radio - Architectures, Systems and Functions*. Wiley, 2003.

[11] Escudero, A., "Location privacy in ipv6 tracking binding updates," in *Proc. of the International Workshop on Interactive Distributed Multimedia Systems and Telecommunication*, Sept. 2001.

[12] The federal communications commission, " Authorization and use of software defined radio: first report and order." *Available HTTP: http://www.fcc.gov/Bureaus/Engineering_Technology/Notices/2000/fcc00430.txt*, last viewed on June 2005.

[13] Fenwick, P., "A new data structure for cumulative frequency tables," *Software Practice and Experience*, vol. 24, pp. 327–336, Mar. 1994.

[14] Frederick, M., "Cellular telephone fraud anti-fraud system," *US Patent 5,448,760*, 1995.

[15] Freier, A., "The SSL protocol - version 3.0." *Available HTTP: http://home.netscape.com/eng/ssl3*, last viewed on June 2005.

[16] GSM, "The GSM security technical whitepaper for 2002." *Available HTTP: http://www.hackcanada.com/blackcrawl/cell/gsm/gsm_security.html*, last viewed on June 2005.

[17] Gupta, V. and Gupta, S., "Experiments in wireless internet security," Tech. Rep. TR-2001-103, Sun Microsystems, 2001.

[18] The gzip website. *Available HTTP: http://www.gzip.org*, last viewed on June 2005.

[19] Holzmann, G., *Design and Validation of Computer Protocols*. Prentice Hall, 1991.

[20] Intel, "Intel wireless trusted platform: security for mobile devices." *Available HTTP: http://www.intel.com/design/pca/applicationsprocessors/whitepapers/300868.htm*, last viewed on June 2005.

[21] The J2ME Specification. *Available HTTP: http://www.wireless.java.sun.com/j2me*, last viewed June 2005.

[22] MacDonald, J., *File system support for delta compression*. PhD thesis, UC Berkeley, May 2000.

[23] Michael, L., Mihaljevic, M., Haruyama, S., and Kohno, R., "A framework for secure download for software defined radio," *IEEE Commun. Mag.*, vol. 40, pp. 88–96, July 2002.

[24] Notare, M., "Wireless communications: security management against cloned cellular phones," in *Proc. of the IEEE Wireless Communications and Networking Conference*, Sept. 1999.

[25] Qi, H., Dapeng, W., and Khosla, P., "The quest for personal control over mobile location privacy," *IEEE Commun. Mag.*, vol. 42, pp. 130–136, May 2004.

[26] Rappaport, T., *Wireless Communcications - Principles and Practice*. Prentice Hall, second ed., 2002.

[27] The software defined radio forum. *Available HTTP: http://www.sdrforum.org*, last viewed on June 2005.

[28] US Secret Service financial crimes division. *Available HTTP: http://www.secretservice.gov/financial_crimes.shtml#Telecommunications*, last viewed on June 2005.

[29] Smith, C. and Gervelis, C., *Wireless Network Performance Handbook*. McGraw-Hill, 2003.

[30] The Trusted Computing Group. *Available HTTP: http://www.trustedcomputinggroup.org*, last viewed on June 2005.

[31] The TCG PC Specification. *Available HTTP http://www.trustedcomputinggroup.org/downloads*, last viewed on June 2005.

[32] Tridgell, A. and Barker, P., "The rsync algorithm," Tech. Rep. TR-CS-96-05, The Australian National University, 1996.

[33] Wagner, D., "GSM cloning." Available HTTP: http://www.isaac.cs.berkeley.edu/isaac/gsm.html, Internet Security, Applications, Authentication and Cryptography Group - UC Berkeley, last viewed on June 2005.

[34] Ziv, J. and Lempel, A., "A universal algorithm for sequential data compression," *IEEE Transactions on Information Theory*, vol. 23, pp. 337–343, Nov. 1977.

VITA

Alessandro Brawerman was born in Santa Maria - RS - Brazil on October 26th 1977. He received his B.S. and M.S. degrees in Computer Science from the Federal University of Parana - Brazil during December of 1997 and May of 2000.

He began his research towards his doctoral degree in the School of Electrical and Computer Engineering at the Georgia Institute of Technology during Fall of 2001. His research interests are in security for software defined radio devices, such as cellphones and PDAs, and security and management of wired/wireless networks.

Mr. Brawerman currently holds a scholarship from Brazil and his research is funded by the Brazilian National Council of Research CNPq. He has published and presented over 10 technical papers. He is an IEEE member and was a Fellow of the Panasonic Information & Networking Technologies Laboratory (2003-2004). He was also the instructor of several undergraduate Computer Science courses in the Federal University of Parana (1998-2000) and in the Computer Engineering School in the Positivo University Center (2000-2001).