Single Sign- on Using Trusted Hardware Background

Souheil Lazghab

Master's Thesis
Master of Science in Information Security
30 ECTS
Faculty of Computer Science and Media Technology
Gjøvik University College, 2007

Avdeling for
informatikk og medieteknikk
Høgskolen i Gjøvik
Postboks 191
2802 Gjøvik

Faculty of Computer Science
and Media Technology
Gjøvik University College
Box 191
N-2802 Gjøvik
Norway

Abstract

The main goal of this thesis is to design and analyze a security protocol for a vulnerable SSO system developed earlier by NISLAB in a previous master thesis and to investigate whether the protocol could enhance the security of the prototype without affecting its usability.

During the protocol design, a substantial part of the work was dedicated to selecting the best cryptographic algorithms that can be implemented in the prototype in order to secure the communication between the mobile phone and the microchip devices. Due to the limited processing power of these devices, the choices of potential cryptographic algorithms for implementation were limited. Most of the algorithms studied in this thesis to secure the prototype, were either subject to some minor changes or used in their weakest form. The limited processing power and memory storage of the microchip device meant that the design of the protocol had to be tuned several times to make it compatible with the hardware's available power of the microchip without affecting neither the usability nor the performance of the SSO prototype.

The protocol developed during this thesis for securing the prototype offers a good level of security given the nature of the tasks that the SSO prototype is expected to perform. More security could only be achieved by using more powerful microchip devices. The work on this thesis resulted in enhancing the security of the SSO prototype by finding some theoretical as well as practical solutions to its major security problems.

Keywords: Single sign-on, Bluetooth, Security protocols, Protocol Design.

Acknowledgements

I would like to thank my supervisor Dr. Chik How Tan, for providing excellent guidance and his precious time. Without the many interesting discussions and his help, I can say that this thesis would not have been the same. I would also like to thank Dr. Einar sekness, Mats byggely and Torkjel Søndrol for providing valuable feedback and help.

Moreover, I would like to thank my brother Dr. Sami Lazghab and my friend Per Vidar Hestness for their support during these last three years. I would also like to give special thank to my father and mother.

Souheil Lazghab, June 30, 2007

Contents

List of Figures

List of Tables

1. Introduction

1.1 Topic

The main goal of this thesis is to design a new security protocol for improving the security level of the SSO prototype developed in an earlier master thesis by NISLAB. Analysis was conducted on the protocol to evaluate whether the protocol is secure enough and whether it represented a practical solution for the vulnerabilities of the SSO prototype or not.

1.2 Problem Description

Nislab have developed an SSO prototype in a previous master thesis. The prototype was a successful one that met all the functionality specifications by a today SSO System. However, because of the nature of the scope of the previous master thesis, the prototype didn't focus on the security issues. The prototype had a number of security weaknesses and was vulnerable to many kind of attacks that could compromise the system and expose its users to many dangers.

1.3 Motivation

The SSO prototype was successful, however, it was very vulnerable to a number of security threats. A considerable amount of work needed to be done before the prototype could be considered stable and secure SSO system. The core of this master thesis is to secure the data streams going under the communication between the two main devices in the prototype, the mobile phone and the PICDEM FS USB Demo board.

The SSO prototype was supposed to implement a device developed by NISLAB. However, this device had some technical problems and was replaced by a PICDEM FS USB Demo board. Both devices are similar because they make use of the same type of microchip. This means that the security protocol developed during this thesis will be relevant for further implementation of the prototype based on the original NISLAB device. The only part of the code that has to be rewritten in case where the original device will be used is the communication module between the Bluetooth and the USB units. Because these units are separate in the Demo board device and they communicate through the RS232 interface. On the NISLAB device, the chip from the USB and the Bluetooth device are connected on the same circuit board. This means that the communication between these chips can be done internally.

The security protocol's task is to secure the sensitive data stored in the record store management (RMS) of the mobile phone device and the Bluetooth channel which the mobile phone and the Demo board device communicate through it. In order to achieve these goals, the protocol should

- Implement a strong and secure authentication mechanism of the user against the sso MIDlet installed in the mobile phone.
- Authenticate the mobile phone and the PICDEM FS USB Demo board devices with each other and finally.
- Provide a good protection mechanism of the sensitive data sent from the mobile phone to the Demo board device.

1.4 Research Questions

In order to develop a reliable protocol design for securing the SSO prototype, the following requirement had to be met:

- The design should include an authentication mechanism that prevents attackers from getting access to the sensitive data stored in the (RMS) of the mobile phone device.
- The design should provide a security mechanism for protecting the sensitive data stored in the (RMS) of the mobile phone device.
- The design should provide an authentication mechanism between the mobile phone and the Demo board devices. The mechanism should be performed prior to the communication and every time the two legitimate devices make an attempt to start a new communication.
- The design should protect the access credentials against man in the middle attacks, i.e. protect against compromising of or tempering with usernames and passwords when they are sent via Bluetooth between the two endpoints devices in the SSO prototype.

Out of the above listed requirements, it was decided to focus on finding practical solutions to the following questions during work on this thesis.

- How the design should protect the sensitive data from the man middle attack?
- How to protect the sensitive data stored in the (RMS) of mobile phone device.
- How the user should be securely authenticated to the SSO MIDlet, which is installed in the mobile phone device.
- How the protocol should authenticate the mobile device and the Demo board with each other.
- How and where should the security protocol be implemented in the design of the SSO prototype without affecting the usability or stability of this prototype.
- What are the alternative solutions in the case the design is not able to provide a good security level due to limited processing power of the devices in the SSO prototype forcing the use of a weak cryptographic algorithms.

1.5 Method

To achieve the goal of this theses an analysis approach is used. This approach is ideal for the type of research questions addressed in this thesis. The following relevant subjects have been reviewed.

- How to design and analyze the security protocols for wireless systems.
- Investigating the technical specification of every device used by the SSO prototype and its related communication and security protocols.
- Investigating the MPLAB IDE and Java developments environment.
- Investigating the security advantages, disadvantages and the best way to implement every cryptographic algorithm that may be used in the security protocol.

1.6 Summary of Claimed Contributions

The contributions of this thesis are as follows:

- Security protocol that should be implemented into the soo prototype.
- Analysis of the different security mechanism provided within the protocol as well as a justification of the major security counter measures and the performance of all the cryptographic algorithms implemented by the protocol.
- Recommendations regarding how and where the security protocol should be implemented in the SSO prototype.

1.7 Scope

The plan was to make our proper implementations for the cryptographic algorithms that will be used by the security protocol and integrate the protocol to the design of the prototype. However, due to the time limitation, both of these tasks were put on hold. Although, these are important tasks, they are not as complicated as the design task, which represents the core of this master thesis. Implementing the cryptographic algorithms or integrating the protocol into the design of prototype should be performed without any technical complication when the theory and the technical feedbacks related to the implementations of the algorithms or the protocol in this paper will be followed properly.

The prototype had to be tested again since the results produced from the previous tests that were conducted on the prototype were outdated. More tests dedicated for investigating the security of the prototype e.g. practical penetration test rather than a large scale test in this step are needed. However, to conduct this test, the protocol to the design of the prototype needed to be integrated.
The security analysis conducted during this work will ensure the integrity of security solutions in this step. In later stages after conducting the practical penetration test, security and user tests that should be dedicated to investigate the stability and usability of the prototype need to be performed. These tests are however, beyond the scope of this thesis, e.g. we could not implement a large scale security test because many other channel in the prototype need to be secured with other different counter measures based on different security approaches that are more related to social engineering rather than security technical attacks. The usability tests where users are supposed to find problems with the user interface and other details of the system implementation in general are not related to the scope of this thesis.

1.8 Outline of Chapters

The rest of this paper is organised as follows:

- Chapter 2 provides the necessary cryptographic technical background for this thesis and introduces other single sign on solutions. It also includes a section about the technical specification of the Bluetooth radio-broadcasting technology along with valuable information about the design of Bluetooth security protocol and its major security weaknesses.
- Chapter 3 provides a technical description of the previous prototype. It also introduces also the security threats that the protocol should eliminate or minimize. This chapter also provides too a recommendation to which parts of the prototype the security counter measures that are provided by the protocol should apply.

- Chapter 4 provides the different design requirements that should be met by the security protocol. These requirements consider in high level the different limitations that the design of the prototype suffered from and which could present a real obstacle for our work in this thesis. This chapter also provides too the methodology, which should be used during the design of the protocol in order to meet the security requirements from section 1.4 of this chapter.
- Chapter 5 provides a detailed description of the prototype. It gives practical solutions for protecting the (RMS) of the mobile phone device along with the main Bluetooth radio-broadcasting channel, which exists between the mobile phone and the Bluetooth/Demo board devices. It also provides too a justification for the choice of these technical solutions.
- Chapter 6 introduces the security protocol designed during this thesis for the soo prototype. It provides explanation along with a full analysis of the different security mechanisms and counter measures implemented within the protocol.
- Chapter 7 provides the technical recommendations with respect to where the protocol should be implemented in the design of the prototype. These recommendations consider all the limitations encountered during the work investigation in this thesis in both platforms used for developing the prototype
- Chapter 8 provides a valuable discussion around the different issues and technical limitations, which we encountered during our work on this thesis. It discusses issues from the previous design and issues that may result from implementing the protocol to the design of the prototype.
- Chapter 9 suggest further work that should be conducted on the prototype in order to be commercialized for the recent future.
- Chapter 10 provides conclusion reached.

2. Preliminaries

2.1 Symmetric Protocols

Encryption protocols can be classified in two classes. The first class includes the symmetric protocols that are based on using secret keys. The second class includes the asymmetric protocols that are based on using public keys. These two kinds of protocols are similar. The only difference is how they deal with the task of providing a shared secret key between the two communicating principals. During this thesis, a symmetric security protocol for protecting the SSO prototype will be designed [01].

The advantages of using symmetric protocols consist in that:

- Most of the symmetric encryption algorithms are faster than the asymmetric one.
- The key schemes of these algorithms are considerably shorter.
- The Symmetric cipher can be used to construct different cryptographic mechanisms such hash function, pseudorandom number generators and digital signature and more.
- The symmetric ciphers produce stronger cipher, that could be used later to produce strongest product ciphers.

The encryption algorithms are classified in two categories, stream ciphers and block ciphers algorithms. Stream ciphers require less processing power and complex hardware circuitry from the hardware than the block cipher. Stream ciphers are usually based on using the symmetric schemes. The encryption function in these ciphers can vary during processing the plaintext. The encryption in these ciphers depends on three factors the key, the plaintext and the current state. The block cipher could run in a stream cipher mode with large blocks by adding a small amount of memory to the block cipher (CBC mode) [01].

Stream ciphers are widely used in the design of the security protocols for systems with limited processing power or when characters of the exchanged message must be processed individually in the receiving principal during the communication. Stream ciphers are ideals for application with high transmission error rate because the error propagation rate in these ciphers is very low or negligible [01].

Key stream in the stream ciphers appears to be random to computationally bounded adversary because it was generated pseudo randomly from a smaller secret key usually known as a seed. Stream ciphers could be synchronous or asynchronous. The difference between these two class is that for the first class the key stream is generated independently from the plaintext message of the cipher text whereas in the second class the key stream is generated out from the previous value of the fixed number of the previous cipher text digits and the key. It is possible to use different function to generate the key stream for the asynchronous ciphers. The synchronous mode is more used because it is able to synchronize itself to continue the decryption in the case where some part of the cipher text is missing [01].

However, the stream ciphers could be very vulnerable to cryptanalysis attacks if the same key is used to encrypt more than one message. In order to overcome this problem, we have to make use of one of the two following approaches:

- The first approach is based on using a single stream or to change the keys for every session. In order to implement this solution, the two communicating principals must share the exact position in the stream from, which the next key stream words are going to be taken for the next session. A common problem with this approach is that the two communicating principals may encounter a communication problem that would result on losing the synchronization and some portion of the data. The two communicating principals will not be able to communicate again only when they will be able to re-synchronize themselves again. In order to reach the exact point of stream where the communication should restart again, the two communicating principals have to use key stream generators, which provide a logarithmic random access property. Fluhrer and McGrew developed the idea of dividing the stream into intervals and use one interval per session.
- In the second approach, the session key is derived from the secret key and an initialization vector that contains the additional amount of data. This vector is not secret and could be transmitted in a clear text. The session key could be derived by simply using concatenation and XORing functions or by hash functions. The hash functions give more secure session key where the concatenation could weaken the security of the session key.

In the following section the three algorithms recommended for implementation in the design of the security protocol developed during this thesis are presented.

2.1.1 AES

The Advanced Encryption Standard (AES) was considered by the NIST in 2000 as the standard algorithm of encryption to be used in protecting sensitive data that belongs to the USA government [02, P 43-46]. The AES has been chosen from among many other competing algorithms e.g. Mars, RC6, Rijndael, serpent and Twofish because it is secure, efficient, easy to use and flexible. The AES has a high performance in software and hardware implementation, supports fast key change because it requires minimal effort for key preparation, need small memory and make use of simple operations to protect against power and timing attacks.

The AES is a block cipher algorithm with variable block lengths. The block length could vary from 128, 192 and 256. It is executed iteratively and each additional bit in the key results in doubling the strength of the algorithm. The AES is a symmetric algorithm since the same key is used for encryption and decryption operations. The key is the only defense mechanism for protecting encrypted data [02, P43-46].

The AES algorithm can be used in a different way to perform encryption. It is very important to use the right method of encryption for the right situations. If the wrong method is used, it could result in making the encryption insecure. AES is very easy to implement in a system, but it should be implemented in the right way for a given situation.

The encryption process is based on 10 successive identical rounds. The result from the previous step is organized in 4-4 byte state matrix. Every rounds of the AES encryption algorithm consists of four transformations, SubBytes, ShiftRows, Mixcolums and AddRoundKey. The AES is executed without concurrent checking during these transformations. During the initial round only the round key is added to the plain text by using the transformation AddRoundKey. In the last round the MixColumns transformation is discarded. These two adjustments proved to be efficient in enhancing the security of the algorithm. The rounds keys are generated from the key of the user by the key expander [02, 43-46].

In order to perform concurrent checking, we need to make use of the XOR-tree to determine the parity of the inputs of every round. The parity should be modified according to the processing steps of the AES algorithm into the parity of the outputs and compared with the actual parity of the outputs of that round. By using the same hardware, the output parity of a current round should be the input parity of the next round [02, P43-46].

2.1.2 RC4

The RC4 is a stream cipher algorithm with a pseudo random bit sequence RC4(IV, K), which depends only on the key K being used and an initialization vector IV, which is generated during the algorithm and XORed to the plaintext P being encrypted. The decryption is achieved by an XOR-operation of the cipher text with the same pseudo random bit sequence or key stream [02, P46-50].

In order to ensure the security of the algorithm, every message is encrypted with a new key stream. The key stream is not identical to any of the key streams used for encrypting the plaintext of previous messages. Further, the initialization vector IV is always mixed with the key to avoid the possibility of two plain texts encrypted by the same IV. If the same IV is used for the encryption of two plain texts P1 and P2, the XOR-operation of the two cipher text C1 and C2 could compute the XOR-operation of the two plain texts P1 and P2. In this case, an attacker doesn't need more than one of the plain text P1 or P2 to compute the other plain text [02, P46-50].

The RC4 algorithm can be used with different key length variables and initialization vector IV. The maximum length of the key is 2048-bits. This will secure the RC4 algorithm against brute force attacks. However, the reduction of the key length to only 40-bits makes the RC4 vulnerable to spoofing attacks. The implementation of the RC4 with 128-bits key lengths make it more secure against these kinds of attacks [02, P46-50].

The RC4 is composed from two parts [04].

- The Key Scheduling Algorithm (KSA) responsible for transforming a random key of 40 to 256-bits size into an initial permutation s included in Sn.
- The Pseudo Random Generation Part (PRGA), which make use of this permutation to generate an output of pseudo-random sequence.

The PRGA generate two variables i and j which value are set to 0. Then, it loops over four operations [04]:

- First, increments the value of i as counter.
- Second, increments j pseudo randomly
- Third, exchange the two values of s pointed to by i and j.
- Fourth, output the value of S pointed to by s[i] + S[j]2.

Every entry of S is swapped a least once (possibly with itself) within any N consecutive rounds and thus the permutation s evolves fairly rapidly during the output generation process.

The KSA part of the algorithm consists of N loops, which are similar to the PRGA round operation. The S is initialized to be the identity permutation and i and j to 0 then the round operation is applied for N times, stepping i across S, and updating j by adding S[i] and the next word of the key(in cyclic order) [04].

The RSA organization claims that the RC4 algorithm is immune to differential and linear cryptanalysis and that the pseudorandom bit generator has no small cycles. The weakness of the RC4 is the method used for combining the key and the initialization vector IV. In order to recover the plain text, an attacker needs only the first octet of the encrypted plain texts. Depending on the generator method of the IV used during the encryption, an attacker needs around between one and four million of plain and cipher text that were encrypted using the same key and different IV [02, P46-50].

2.1.3 Hash Functions

Hash Functions are used to sign the cotenant of a stream of data. It takes as input a string of bytes with different length and converts it to a fixed length output. The function is deterministic and efficient but random too. The minimum change submitted on the input of this hash functions should result in a change too on the output of these functions [05].

The hash function has the following rules [06]:

- It should be very difficult to find a corresponding input m for a specific output h. h = H[m] where H is the hash function.
- Considering that we have an input m1, it should be very difficult to find a matching input m2 where H[m1] = H[m2].
- It should be very difficult that two different messages m1 and m2 will produce the same hashed value H[m1] = H[m2]

Hash functions are typically used to:

- Sign a document, first the document should be hashed then the hashed value will be encrypted with the private key of the sender so the recipient could be sure about the identity of the sender.
- Check whether a file was modified.
- To avoid getting access to the clear text of the passwords, so the hashed values of the passwords are stored instead of its clear text.

Hash functions can be classified into two classes:

- Unkeyed hash functions, which recommend a single message.
- Keyed hash functions, which recommend two distinct inputs, a message and a secret key.

Hash functions must have two fundamental proprieties. First, it should be able to perform good compression and second it should be easily computed.

The hash functions are also classified in two major classes depending on their functionalities:

- *Manipulation detection codes MDC*, This class of hash functions belongs to the unkeyed category of hash functions. They provide a representative image or *hash* of a message. The end goal is to provide in conjunction with additional mechanisms (see x9.6.4), the integrity of the data that is required by specific applications.
- *The message authentication codes* (MACs), this function is classified under the keyed hash functions category. These assure the source and the integrity of the message without any additional mechanisms. They require two inputs the message input and a secret key.

SHA-1:

The Secure Hash Algorithm SHA-1 was developed by the U.S. National Institute for Standards and Technology NIST for certain U.S. federal government applications [01].

- The hash-value is 160-bits, with five 32-bits chaining variables.
- The compression function is composed of four rounds, the same step functions used in the MD4 are used in the SHA-1 but in different order.
- The compression function process each 16-word message block and expanded it to an 80-word block message. Each of the last 64 of the 80 words is the XOR of 4 words from earlier positions in the expanded block. These 80 words are then input one-word-per-step to the 80 steps.
- The core step for the SHA-1 should be modified as follows: the rotate is a 5-bits constant rotate; the fifth working variable should be added into the result of each step; the message words from the expanded message block are accessed sequentially then and C is updated as B rotated left 30-bits.
- The SHA-1 uses four non-zero additive constants.
- The SHA-1 uses big-endian byte ordering to convert between streams of bytes and 32-bitwords. There are no particular advantages in choosing big-endian or little-endian. The choice of one function over the other depends on the computer platform of the SHA-1 developers rather than the design decision.
- The SHA-1 160--bits hash-value is more secure against brute-force attacks than 128-bits hash-value.

2.2 Related Work

2.2.1 The Single Sign on Solutions Available for Today

The single sign on systems available for today are classified into four main categories. The SSO implementation for each of these categories is introduced in the section below [52].

2.2.1.1 The Local Pseudo Single Sign on Solutions Category

This category of single sign on systems is considered to be a password manager that acts as a digital safe. The SSO components are located on the user's machine. The access credentials in these systems are stored in an encrypted database. The database is protected by a master password. The master password acts as an authentication and encryption key at the same time. The generation of this master password is dependent on each system's implementation. Some of the implementations provide password checking and secure generation of passwords. All the developed systems that belong to this category do not support automatic log inn of the user [07].

The security of the local pseudo SSO system depends only on the security of the master password. However, this password is not always chosen in a secure way because many of the users would chose guessable passwords that are easy to remember. Once an attacker succeeds to steal the master password or compromise it, he will be able to get access to all the sensitive data that are stored in the system.

Most of the local pseudo SSO systems are considered to be a partial mobile solution of the mobile single sign on systems. This due to the fact that most of the implementations of this category don't provide automated login and are designed to act as a digital safe. There were many implementation dedicated to run over smart cards, USB sticks devices e.g. Password Director [08]. The user in this system implementation could not use his device only in a computer where software belongs to the system should be installed. If the software is not installed, the user could not use the device acting as digital safe. Other implementations of the system are dedicated to be run on the mobile phone devices where a user can have all his passwords protected in a digital safe on his mobile phone e.g. SecureWord.Mobile [09]. However, some of this implementation allows communication with similar product on a computer and some of the implementations provide even an automated logins for its users but only to web forms e.g. MobiPassword [10].

2.2.1.2 The Proxy-Pseudo Single Sign on Solutions Category

The proxy based pseudo SSO category is a web based system [52]. The SSO components are residing on an external machine that acts as a proxy for the SSO system. The authentications in these systems are performed between the proxy and the server provider not between the user and the server provider. This category is similar to the SSO implementation designed to run on single PC. The main differences are:

- In the previous implementation the local machine, which the user is using needs to get access to the sensitive data while in this implementation the local machine does not have any access to the access credential of the user.
- The passwords and the encrypted sensitive information are protected in an online proxy server that every user should have access to by using a master password and the login procedure could be automated.

Similar to the Local Pseudo SSO, the security of this SSO category is based on the security of the master password. If an attacker managed to get access to the master password or compromise it, he will be able to retrieve all the encrypted protected data residing on the proxy server. The most known implementation for this category are the Online Password Manager [11] and MyPasswordManager [12]

In order to avoid the compromising of the master password, an improved implantation of the proxy based pseudo SSO system was developed by Pashalidis and Mitchell [13]. The solution is based on using a trusted proxy, which performs the real authentication for the user to the targeted web site. Every user in this solution must store a copy of their access credentials into the trusted proxy. Another solution proposed by Samar is based on using centralized cookie server to perform the authentication of the user to the system [14]. This solution is relevant only for web application since it is based only on cookies.

2.2.2 The Local True Single Sign on Solutions

These single sign on systems are based on using the trusted computing platform alliance (TCPA) platform, which requires the implementation of a public key (PKI) and a trusted component [15].

2.2.2.1 The Proxy-True Single Sign on Solutions

This proxy based true SSO system is based on using a proxy web based solution. Many implementations of this category are available in the marked today:

- The Microsoft Passport implementation, which makes use of an encrypted cookie as ticket for each user to authenticate them to the system [16].
- The Pashalidis and Mitchell implementation, which is based on authenticating the user to the SSO system by using GSM/UMTS as an authentication service provider (ASP) [17].
- The Protocom SecureLogin implementation, which is based on authenticating the user to a centralized manager [18]. These systems provide a good access control system and offer its users different types of login e.g. login to web forms, database servers and terminal emulators. This system offer to its users an automated login as well as a good protection of their sensitive data because it resides in a central server, which can be well protected. The system allows its users to use two methods of authenticating the master password, which is the default method or a more secure authentication method which is based on using a smart card with PIN code.
- The SSO implementation based on Kerberos V where a user authenticates himself to the system by using a ticket issued from the key distribution center (KDC). It makes use of the secure Kerberos authentication protocol [19].
- The Satoh and Itoh implementation, which is based on a totally different method of authentication than the four previous one [20]. The system in this case makes use of dynamic tokens instead of static tokens. This approach makes use of a server for issuing valid dynamic tokens to the user of the system.

The best implementations for the SSO system are the ones developed for the business community e.g. Protocom SecureLogin, Utimaco Safeguard and the Kerberos based implementation. However, most of these implementations are very expensive solutions and requires extra payments for additional service. They require specific hardware processing, memory power and complicated configuration procedures.

These four implementations category of the SSO system could be divided in two major families:

- The pseudo SSO family.
- The true SSO family.

The major differences between these two families are in the true SSO implementation. The user is authenticated against an authentication Service Provider (ASP). In the pseudo SSO implementation, the access credentials are passed to the service provider. Another difference between the two implementations is that in the proxy based true SSO system, the ASP is residing on an external server, whish acts as a bridge between the user and the server provider while in the local true SSO implementation, this process is based on using trusted component located in the user system and, which acts as an ASP.

2.3 The Security Issue in the Bluetooth Protocol

The Bluetooth technology was developed by the Bluetooth Special Interest Group (SIG) in 1998. It is sometimes referred too as the IEEE 802.15.1, which means the Wireless Medium Access Control (MAC) and Physical Layer (PHY) specification for the Wireless Personal Area Networks. The Bluetooth technology is a radio based communication technology. It is an open

industry standard for the unlicensed short-range radio communication of voice data. The Bluetooth radio communication allows for an ad hoc networking within the range of 10 up to 100 meters. The Bluetooth operates in the 2.4 GHz ISM frequency band on 79 channels by using the frequencies f = (2402 + k) MHz where k = 0, … , 78.[21]

The Bluetooth offers its users an asynchronous asymmetric connectionless data transmission up to 723.2 kbps for one direction and 57.6 kbps in the other direction or an asynchronous symmetric connectionless data transmission with up to 433.9 kbps for both directions. The Bluetooth also offers synchronous connection oriented channels with each up to 64 kbps. The voice coding is performed via Bluetooth by the PCM (Pulse Cod Modulation) or CVSD (Continuous Variable Slope Delta Modulation) [21].

The Blue Tooth technology was developed to replace the cable communication in a small area like office and buildings environment. The Bluetooth protocol allows establishing a wireless personal area network (PAN) by allowing two or many Bluetooth devices to connect with each other if they are located within the vicinity of each other's. The Bluetooth covering area range is up to 10 meters [21].

The Bluetooth devices can connect to each other as a master or a slave device. The first device that will be activated should acts as a master device, while the others, which join the connection later are considered slave devices. During the connection the master device will always try to discover any newly activated Bluetooth device in its radio-broadcasting range. The broadcasting range depends on which category the master device belongs to. However, the Bluetooth devices could be activated in two modes, discoverable and non-discoverable mode. Only Bluetooth devices configured to discoverable mode are detected by the master device. The Bluetooth protocol starts first the pairing procedure between the master and new slave device. The pairing process is a mechanism based on the challenge response protocol where the two Bluetooth master and slave perform a mutual authentication. The authentication is based on using a unit key, which consists of 48-bits that represents the Bluetooth address and a random number. The unit key is exchanged between the two Bluetooth devices performing the pairing without any kind of protection. This unit key is combined then with a 4 digits PIN value randomly generated by the user who possesses the Bluetooth device to generate a link key. The PIN value is identical for both Bluetooth devices involved in the pairing process [21].

There are three security modes that the Bluetooth security protocol provides [21]:

- The insecure mode. In this mode the Bluetooth device doesn't initiate any security mechanism when starting a communication with other Bluetooth devices. However, it responds to authentication request if it was recommended from other Bluetooth devices.
- The service level security. In this mode the selection of the security mechanism is determined form the trust status of the involved Bluetooth device in the communication and the service application level. The trust status could be trusted or not trusted. The security procedures could only be established if a request for establishing a connection is received to the targeted Bluetooth device.
- The link level security mode. In this mode the authentication of the devices to each other are mandatory before starting any transfer of data. The encryption is however, optional.

The second step in the Bluetooth protocol is the SDAP service, which is used to find out what kind of a service profile is supported by the other Bluetooth device to make a profile for the communication between the two Bluetooth devices [21].

The PIN value represents a weak point in the mutual authentication process described above. This is due to the fact that this value is usually used as a default with 0000 digits as a value to overcome the limitation where in many situations the two party that are involved in the communication don't know each other and they could not agree on a fix value for the PIN prior to the communication.

The Bluetooth protocol is vulnerable to many attacks because [21]:

- The data used during the authentication of both devices are sent without any kind of protection.
- The encryption is performed only after the devices are authenticated, which resulted in making the protocol vulnerable to key replay attacks.
- The user authentication is not supported in the Bluetooth protocol.
- The security protocol is not mandatory in the Bluetooth protocol. It is left optional to the user of the Bluetooth device and it could be switched off completely, which means that all the data exchanged between the two users will be sent without any protection.
- The PIN code is short and is vulnerable to brute force attack.
- The SAFER+ proved to have some weaknesses.
- The encryption algorithm used in the protocol E0 proved to be insecure. It was possible to recover the initial state of E0 and the encryption key.

There are many serious successful attacks that were documented during the last period against the devices communicating through the Bluetooth protocol. The most serious attack that is related directly to the SSO protoype used in this thesis is the BlueSnarfing [22] attack. Previous experiments showed that most of the mobile phone devices that support Bluetooth are vulnerable to this attack when they are configured to be in the discoverable mode. Additional experiments conducted later by the same group of researcher confirmed that some of these phones are vulnerable to this attack even when they are configured in undiscoverable mode. The BlueSnarfing attack is considered to be a ruthless attack because it allows an attacker to access all the access credentials stored on the (RMS) of the mobile phone device without being discovered from the user of the SSO prototype.

The second attack that is directly related to the prototype and proved to be a serious attack is the attack dedicated to exploit the trust weak point in the Bluetooth protocol as explained above in this section. The attack makes use of the trust established between the two Bluetooth devices. During this attack, an attacker tries to get access to the key exchange data that are exchanged between the two devices without any kind of protection. Once the attacker gets access to these data, he could use it later to generate the right PIN code in the case where the devices don't use a simple default PIN key. This is due to the fact that these default PIN keys are usually set to 0000. Many documents show how to conduct a brute force attack in order to crack the PIN key. Cracking this key is not a difficult task for today's technology because it is considered as very short key with only 4 digits. The Bluetooth protocol doesn't perform any checking for the integrity of this PIN key, which is considered to be a very critical key since many other keys are directly derived from this PIN key [23].

Using a short PIN will make the Bluetooth communication channel, which is the main channel in the prototype very vulnerable to two serious attacks that belong to the man-in the middle attack category:

- The relay attack, where an attacker tries to impersonate two legitimate principals by posing him self as the second legitimate principal to every one of them. The principal

here are the mobile phone and the Bluetooth/Demo board device. Then, the attacker starts to record all the data and forward it to both direction of the communication [24].

- The packet modification attack. This attack is based on the same method as the replay attack. The only difference is that instead of only listening and recording the information sent between the two communicating principals, the attacker will temper with it and perform some modification to it [23].

The Bluetooth protocol is used as the technology of choice to assure the communication going between the two main devices in the SSO prototype, the mobile phone and the Bluetooth/Demo board:

- The first reason for choosing the Bluetooth was because the Demo board was the only device that matches the original device developed by the NISLAB to be used in the SSO prototype. Both devices are Bluetooth based technology.
- The second reason was that most of the mobile phones today support the Bluetooth radio based communication.
- The third reason is that the SSO MIDlet is a java-based platform, which supports the Bluetooth communication through its Bluetooth API.
- The fourth reason is that the Bluetooth is one of the newest and updated wireless technologies that is used in many area.

Conducting an attack against a Bluetooth device is not a sophisticated task, there are many hacking tools and softwares available for everybody that could perform serious attacks against the Bluetooth devices such as:

- The Bluestumbler used for monitoring and logging different Bluetooth devices.
- The Bluebrowse, which used as a port scanner.

Table 1: Security Issue of the Bluetooth Protocol

	Security Issue or Vulnerability	Remarks
01	Strength of the challenge response pseudo random generator is not known	The Random Number Generator (RNG) may produce static number that may reduce the effectiveness of the authentication scheme.
02	Short PINS are allowed	Weak PINs, used for the generation of link and encryption keys, can be easily guessed. Increasing the PIN length in general increases the security. However, people have a tendency to select short PINs.
03	An elegant way to generate and distribute PINs does not exist.	Establishing PINs in large Bluetooth networks with many users may be difficult. Scalability problems frequently yield security problems.
04	Encryption key length is negotiable.	The Bluetooth SIG needs to develop a more robust initialization key generation procedure.
05	Unit key is reusable and becomes public once used.	Use a unit key as input to generate a random key. Use a key set instead of only one unit key.
06	The master key is shared	The Bluetooth SIG needs to develop a better broadcast keying scheme
07	No user authentication exists	Device authentication only is provided. Application level security and user authentication can be employed.
08	Attempts for authentication are repeated.	The Bluetooth SIG needs to develop a limit fiture to prevent unlimited requests. The Bluetooth specification requires a time out period between repeated attempts that will increase exponentially.
09	E0 stream cipher algorithm is weak	The Bluetooth SIG needs to develop a more robust encryption procedure.
10	Key length is negotiable.	A global agreement must be established on minimum key length
11	Unit key sharing can lead to eavesdropping	A corrupt user may be able to compromise the security between (gain unauthorized access to) two other users if that corrupt user has communicated with either of the two users. This is because the link key (unit key), derived from shared information is disclosed.
12	Privacy may be compromised if the Bluetooth device address (BD_ADDR) is captured and associated with a particular user.	Once the BD_ADDr is associated with a particular user, that user's activities could be logged, resulting in a loss of privacy.
13	Device authentication is a simple shared-key challenge-respons.	One-Way-Only challenge-response authentication is subject to man-in-the-middle attack. Mutual authentication is required to provide verification that users and the network are legitimate.
14	End-to-end security is not performed.	Only individual links are encrypted and authenticated. Data is decrypted at intermediate points. Applications software above the Bluetooth software can be developed
15	Security services are limited.	Audit, non-repudiation, and other services do not exist. If needed, these can be developed at particular points in a Bluetooth network.

The table is taken from NIST_SP_800-48.PDF [21].

3. Describing the SSO System

3.1 Describing the Previous System

The SSO system was developed to offer a better solution for the issue of password management [52]. The main goals were:

- To solve the issue of the repeated authentication task, which a user should perform every time he needs to log inn to one of his accounts on the internet.
- To offer a better and secure way of managing a large number of passwords.

The SSO prototype was a successful prototype, which met many of the requirements that an SSO system should meet. E.g.

- The prototype succeeded in providing an automated login procedure not acting only as a simple digital safe system.
- The prototype was user-friendly, effective and intuitive.

In order to meet the mobility requirement, the prototype makes use of a PICDEM FS USB Demo board and a mobile phone device. The Demo board device acts as a keyboard emulator, which means when this device is plugged into a computer, the operating system of the computer will considers it as a normal keyboard and will automatically load a generic keyboard driver. This design concept is platform independent and does not need any pre-installation procedure of any software.

However, the prototype partially failed to be a mobile system due to the inappropriate size of the Demo board device used as a key board emulator even it met the requirement to be mobile in such way that could be used on different computers and granting a continuous access to the access credential by storing them in the (RMS) of a mobile phone device.

The access credentials are sent over the Bluetooth to the serial port of the Bluetooth™ Serial Port Plug - F2M01C1 (f2m) as key codes. The f2m device forwards the access credentials then to the Demo board device connected to the PC with a standard USB cable through a null modem adapter. This step is needed to cross the TX (Transmit) and RX (Receive) signals because the f2m and the Demo board have both female RS232 connectors.

The prototype failed totally in protecting the access credentials, which are saved in the (RMS) of the mobile phone device or sent from this device through the Bluetooth radio-broadcasting channel to the Demo board device. The only security mechanism that was provided for the SSO prototype was a simple password authentication function based on using standard string matching function and implemented in the SSO MIDlet, which is installed in the mobile phone device. The authentication function was not performing correctly. A user can still log inn to the SSO MIDlet even if he provides the function with a wrong password.

3.1.1 Hardware Technologies

In this section, the different devices that compose the hardware part of the soo prototype are presented. The overall design of the SSO prototype is shown in figure 1.

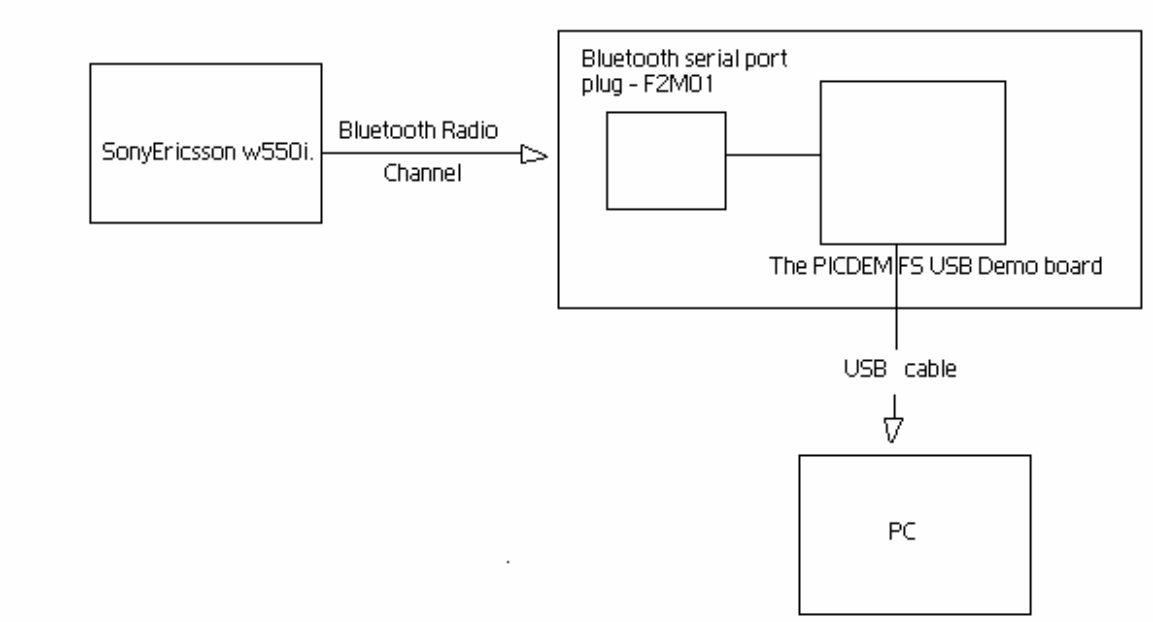

Figure 1: shows a Diagram of the Original SSO System

3.1.1.1 The PICDEM FS USB Demo board

The PICDEM FS USB Demo board device is produced by the Microchip Company. It belongs to the PIC18F4550 category, which supports a full speed USB and offer RS232 serial communication [25]. This device acts as a keyboard emulator in the SSO prototype. Its technical specifications are listed below:

- 48 MHz operating speed.
- 32 Kb of Enhanced Flash memories.
- 2 Kb of RAM.
- 256 bytes of data EEPROM.
- Full speed USB 2.0 interface.
- 20 MHz crystal.
- Serial port connector.
- Connection to the MPLAB ICD 2 in circuit debugger.
- 2 LEDs for status display.

3.1.1.2 The Mobile Device

The mobile phone used in the prototype is a SonyEricsson w550i. The SSO MIDlet developed in the previous thesis is platform independent. It requires only a mobile device that supports Java MIDP, Bluetooth and Java Bluetooth APIJSR-82.

3.1.1.3 The Bluetooth Serial Port Plug – F2M01

This is a Bluetooth device based on the RS232 interface. It plays the role of an intermediate in the SSO prototype [26]. It resides between the mobile phone and the Demo board keyboard emulator devices. The access credentials are sent from the mobile phone to the Demo board through this device by using the Bluetooth channel. The access credentials sent by the mobile phone are received in this device in the form of key codes.

3.1.2 Software technology

The software used for developing the prototype is fully documented in the previous thesis [52]. In this paper only the software directly related to our work for investigating whether the security protocol is practical enough to be implemented in the prototype are listed. The list of softwares used to program the Demo board device are given below:

- MPLAB IDE v7.30 is used for programming the microchip. It is originally used for writing assembly codes. It is a windows based program. It offers a lot of useful tools that helps during the developing process. The IDE supports a lot of hardware programmers and debuggers.
- The C18 compiler v3.02. This is a C compiler for microchips that belong to the category of PIC18, which the PICDEM FS USB Demo board device belongs to [27]. The IDE supports the integration of C18 compiler. The C18 supports the standard ANSI C.

The MPLAB ID supports the ANSI C if the C18 is integrated to it.

The main software that used to program the SSO Java MIDlet is the NetBeans IDE 5.9. [28] In addition to the editor, the following software packages were used [52]:

- The Java 2 Standard Edition (J2SE) v1.4.2 or later releases [29]. This is the basic development kit for the Java 2 platform. It is used for running the Java complier (javac) and for creating the Javadoc and the Java archive (JAR) files.
- The Connected Limited Device Configuration (CLDC) v1.1. Package, which contains a number of Java libraries [30]. These libraries are used in to the mobile phone development environment. They are very limited and don't provide any security functions.
- The Mobile Information Device Profile (MIDP) v2.0, which is a Java runtime environment for the mobile device [31]. It runs on the top of the CLDC.
- The J2ME Wireless Toolkit, which is used for managing the J2ME projects and for simulating the behavior of these project on different mobile devices platforms [32].
- The Bluetooth API JSR-82 used for integrating the Bluetooth radio-broadcasting functions into the new developed application [34] [35].
- The NetBeans Mobility Pack needed for adapting the Bluetooth API with the NetBeans Package [33].

Most of these packages are provided free of charge on the net by the sun organization. Some of these software also have limited security libraries and functionalities.

3.1.3 Design of the SSO Prototype

The design of the prototype is composed of two main parts that are based on two different technologies and development environments, the design of the Demo board device and the Java MIDlet.

3.1.3.1 Design of the Bluetooth/USB Demo Board Device

Bluetooth/USB devices are intermediate devices in the prototype that act as a keyboard emulator and forward the data sent from the mobile phone to the PC. These devices receive the data sent from the mobile phone through a Bluetooth interface and send it to the PC through a normal USB port. The keyboard emulator, which is the Demo board device is connected to the PC with a standard USB cable. The device informs its user about its current state through two LED lights. There are two possible states for the Demo board device:

- Ready to receive data. In this state the LED lights should blink alternately.
- Device not well configured. In this state the LED lights will not blink alternately.

The Demo board device goes through an enumeration process where it gives the PC to which it is connected some information about itself such e.g. it is a Human Interface Device (HID) and some information about the minor class Generic Desktop Keyboard [11]. The keyboard emulator sends to the operating system of the PC a device descriptor, which gives information about the length and the format of the data packages that will be sent during the communication from the Demo board device to the PC. The HID specification recommends an eight bytes length for the report that contains the key strokes [36].

The communication process of the prototype is composed of three major steps. These are:

- The first step, after the enumeration process is performed correctly, the Demo board device starts the USTART interface, which communicates with RS232 COM port. The same port is used for the connection of the f2m device. The communication process in this prototype is a one-way communication process. The communication begins when the MIDlets starts sending data over the Bluetooth through the serial port profile protocol, which is responsible for emulating the RS232 communication.
- In the second step, the Demo board starts receiving the data blindly because it is not able to distinguish whether the received data is emulated serial traffic or normal serial traffic. This is because the data is sent over the RS232 interface. The MIDlet precedes sending data to the Demo board device in one byte mode a time, in the other side the Demo board, which is the receiver device receives only one byte a time. For every received byte in the RX pin of the Demo board device, the interrupt function is invoked.
- The third and last step of the communication process is when the numbers of bytes in the buffer of the Demo board reaches eight or more, it places them in the input report. When the report is ready, the Demo board sends it to the PC, which displays it on the screen. This process is executed iteratively until the buffer of the Demo board is empty.

3.1.3.2 Design of the Java MIDlet

The MIDlet is the second part of the prototype. It is the part where the user of the prototype interacts directly with the system. The MIDlet is designed to help users mange their different access credentials. Every access credential belongs to one account. The MIDlet is user friendly and allows its users to create, alter, delete or store an account easily.

The MIDlet asks for the authentication when it is started up for the first time. However, the function in the prototype was implemented primitively and partially because the user is still able to go forward to the main menu even if he does not provide any password or a wrong password. The users can also change their passwords during the authentication process. After the password checking procedure, the MIDlet prompts the user with the main menu from which a user can create new account or choose an existing one.

A full description of the design of the MIDlet is available in the listed reference [52].

3.2 The Security Weaknesses of the SSO System

The SSO prototype is composed of many principals:

- The Mobile device with the SSO MIDlet.
- The Bluetooth/ PICDEM FS USB Demo board device.
- The computer.
- The user.
- The firmware provider for the Bluetooth/ PICDEM FS USB Demo board device.
- The MIDlet servers where new updated SSO MIDlet can be downloaded from.
- The firmware provider for the mobile phone.
- And the other Bluetooth enabled devices.

The prototype is vulnerable to many kinds of attacks because of the big number of principals involved in this prototype. These attacks have different backgrounds, technical and social engineering attacks.

The attacks that are related to the subject this thesis are of technical background and target the following channels:

- The channel between the SSO MIDlet and the Bluetooth/Demo board device.
- The channel between the user and the mobile phone device.
- The channel between the Demo board device and the PC.

Protecting these channels will ameliorate the overall security level of the prototype. The other vulnerable external channels are out of the scope of this thesis. These should also be protected to avoid many attacks. Vulnerable external channels are listed below:

- The channel between the firmware provider for the Bluetooth/Demo board device and the purchased device used in the prototype.
- The channel between the SSO MIDlet server and the mobile phone device.

- The channel between the firmware provider of the mobile phone device and the purchased phone used in the prototype.

3.2.1 Practical Attacks

This section is based on the security analysis conducted during the previous thesis [52]. The security analysis was based on the adversary modeling method [50]. Some of the reviewed attacks were not totally related to the subject of this thesis because they were dedicated to external channels that are out the scope of the security protocol.

Only the relevant attacks that are directly related to the security weakness that the security protocol should provide a practical solution for are introduced.

Attack 01

The First attack is based on monitoring the Bluetooth radio channel between the Bluetooth/Demo board device and the SSO MIDlet. The Bluetooth connection is protected by a standard encryption connection that proved to be vulnerable to many cryptanalyst attacks due to the weak cryptographic algorithm used in the standard security protocol of the Bluetooth communication [37] [38] Due to the weaknesses of the Bluetooth standard encryption scheme, an attacker could just drive a cryptanalyst attack on the communication between the Bluetooth/Demo board and the mobile phone devices to retrieve the access credentials [39].

Previous research also proved that the paring protocol used for the Bluetooth devices are vulnerable. If The PIN code is about 4 digits, an attacker can compromise the PIN and eavesdrop on the communication between the Bluetooth/Demo board and the mobile phone devices.

The PIN used in the prototype was a default PIN. The previous developers of the prototype had to make this choice to overcome the problem of installing extra software to the PC connecting with the Demo board device in the case where a stronger customized PIN will be used. Binding a user to a specific PC makes the prototype not mobile any longer. In this case the attackers have only to find out the default PIN for decrypting the Bluetooth communication and retrieve the access credentials. Compromising the standard PIN is not a complicated task for an attacker with some cryptographic skills.

This attack is dedicated to the Bluetooth communication channel. During this attack, the attacker intercepts the communication. It is considered to be practical because in order to perform it successfully, the attacker only need to:

- Be located him self within the range of the targeted Bluetooth device that belongs to the prototype.
- To have access to sophisticated radio equipments for intercepting and processing the Bluetooth signals sent from the mobile phone to the Demo board devices.

Attack 02

The second attack is based on monitoring the Bluetooth radio channel by configuring tempered Bluetooth device with the same name as the Bluetooth device used in the prototype, which is “SSO device”. By doing so, the attacker could trick the system and present his device as the legitimate device instead of the real one. The SSO MIDlet first performs a search for the

Bluetooth device with the “SSO device” name. Because there is no any kind of authentication between the SSO MIDlet and the Bluetooth device, the SSO MIDlet will communicate with the first device labeled with the “SSO device”. This Attack is based on getting the right name of the Bluetooth device and the default PIN code. The attacker needs the default PIN code to perform the pairing task explained in the previous attack with the SSO MIDlet. During the attack, the fake device acts as Trojan horse for the SSO MIDlet. An attacker could easily get access to the access credentials sent between the Bluetooth/Demo board and the mobile phone devices. However, this attack could not be stealthy over a long period because the user will discover that there is something wrong with the connection when he will not be able to log into the chosen account. Performing this attack successfully allows an attacker to get only a set of access credential that belongs to one account. It is possible that the user will try another account assuming that there is something wrong with the first one when he figures out that he is still not logged inn. Or he will be skeptical and think about that he was attacked and will abort the connection with the Bluetooth/Demo board device and change the access credential of that account before he will make a new attempt for logging in. The reaction of the user depends on his level of awareness to security issue.

This attack is practical. An attacker intercepts the communication. He only need:

- To configure the name of his/here Bluetooth device to the same name of the Bluetooth device used by the prototype.
- To compromise the default Bluetooth PIN code that belongs to the Bluetooth device of the SSO prototype.
- To locate him self within the range of the targeted Bluetooth device that belongs to the SSO prototype.

Compromising the Bluetooth PIN is not complicated task for an attacker with a good skill.

Attack 03

This is based on controlling the Bluetooth device. During this attack, an attacker makes attempts to connect to the Bluetooth/Demo board device. This Attack could be considered as a replica to the previous attack but in the opposite way. In order to perform the attack, an attacker:

- Has to establish a connection to the Bluetooth/Demo board device. In order to do so, he should crack the default Bluetooth PIN code used by the prototype.
- He can use the Demo board device that act as a keyboard emulator to remotely control the PC that the Demo board is connected to through a standard USB cable.
- He could use some techniques to guess the access credential of the user such as entering key strokes remotely to the PC connected to the prototype by instructing the Demo board device to convert ASII characters to key codes. The attacker could download a key logger as a Trojan horse to the targeted PC or could access some useful information like internet cookies residing on the targeted PC for later use.

During this attack, the attacker can modify the data sent during the communication. However, this attack is a little bit complicated because many issues have to be taken into consideration when an attacker conducts the attack against the prototype.

- First, the limited processing and memory power of the Demo board device.

- Second, the user will notice many strange events on the PC that may make him react by disconnecting the Demo board from the PC, closing the PC and abort the whole log inn operation. Doing so will oblige the attacker to give up and stop the attack.
- Third, the SSO user should leave the PC open without logging off from it.
- Fourth, the attacker has to locate him self within the range of the Bluetooth device form 10 to 100 meters.
- Fifth, the Bluetooth/Demo board has to be connected to the targeted PC.

In order to drive this attack, the attacker needs a program installed in the Demo board device for converting the ASII codes to emulated codes in addition to the same devices recommended for the first and the second attacks.

Attack 04

This attack is based on controlling the computer, which the Demo board is connected to through exploiting some software vulnerabilities installed in the PC. In order to perform this attack, the attacker must have the possibility to control the PC e.g. installing a key logger program to the PC, which monitors the data sent from the Demo board to the computer. By doing so the attacker will be able to:

- Retrieve any piece of information sent from the SSO MIDlet through the Bluetooth/Demo board devices to the PC.
- Write e.g. injecting some malicious codes to the prototype.

This attack is considered to be very practical attack. During this attack, the attacker intercepts the communication. In order to perform this attack, the attacker has to use of port scanners, Trojans horses and password crackers programs. This is considered as a very practical attack.

Attack 05

This attack is based on getting access to the (RMS) of the mobile phone device where the SSO MIDlet saves all the sensitive information that belongs to the user of the prototype. An attacker can get access to the mobile phone by stealing it or accessing it when its owner is away. If the attacker manages to get access to the mobile phone while the SSO MIDlet is running, he will be able to access the sensitive data with zero effort. In the worst case, the attacker has to crack the master password to get access to the accounts and its relative access credentials. In order to crack the password in a short amount of time, the attacker could benefit from better processor and memory power by downloading the content of the (RMS) of the mobile phone to a powerful laptop and crack the main password with powerful password cracker software. The main password presents the only defense wall for the SSO prototype we are working on in this thesis.

In order to perform such an attack successfully, an attacker should

- Manage to get access to the mobile phone device.
- Be acquainted with using password-cracking software.
- Possess of a powerful computer laptop or PC.

Most of the attacks described above represent a high threat to the user of the prototype. They recommend only medium effort from the attackers. The amount of effort needed for conducting these attacks depend on how good the skills of the attackers are. These attacks could be ruthless

and could compromise the security of the prototype. Most of them are based on exploiting some well known technical weakness that exists in the security protocol of the Bluetooth.

The SSO prototype is vulnerable to other kinds of attacks that are based on the cooperation of two or more independent attacker. Only the technical attacks that are directly related with securing the three vulnerable channels that the design of the security protocol must protect in the prototype are discussed. The other attacks that are based on social engineering will not be discarded because they are out of the scope of this thesis.

However, there is only one technical based attack that requires the cooperation of two attackers. The attackers should possess a Bluetooth/Demo board device. This attack consists in the following main steps:

- First, the first attacker tricks the prototype user to start a connection with him. Once the connection is established the first attacker records all the access credentials sent to him from the user who thinks he is the legitimate Demo board device in this connection.
- Second, the first attacker then forwards this information to the second attacker, who is standing by through another Bluetooth radio-broadcasting channel.
- Third, after forwarding these access credentials to the second attacker, he could connect to the legitimate Bluetooth/Demo board device, which belongs to the prototype and which is connected to the PC and start sending these access credentials that will make him able to access the accounts belonging to the user of the prototype.

During this attack the user of the prototype will not be able to discover any strange activities. The security protocol could protect against such attacks by encrypting the Bluetooth channel between the mobile phone and the Demo board device. Because the Bluetooth/ Demo board device could not decrypt the access credentials sent from the mobile phone only by using a secret encryption key. This key is not exchanged between the mobile phone and the Demo board device. It is installed in the fixed memory of the Demo board.

All the attacks we introduced above could be neutralized or minimized by implementing a security protocol to the prototype. More technical explanation will be provided in the following section about these technical counter measures for eliminating all these attacks introduced in this section.

3.3 Which Part that Needs To Be Protected

Based on the previous section it is conclude that the Bluetooth radio communication channel is vulnerable and needs to be protected because the encryption protocol used as a default in the Bluetooth radio protocol communication proved to be insecure from the cryptanalyst community. Many successful reported attacks were performed on this protocol in the recent years. The vulnerability of this encryption protocol resulted in making the main communication channel in the prototype residing between the Bluetooth/Demo board and the mobile phone devices vulnerable to the attacks discussed early in the previous section.

Another issue that further worsens the situation and makes this communication channel very vulnerable is that the Bluetooth PIN used in the SSO prototype. This is a default 4 digits PIN code that could be easily cracked as described early in this paper, even if the PIN code is used only one time during the pairing of the Bluetooth with the SSO MIDlet during the first time. This is due to the fact that the SSO MIDlet keeps the URL connection of the Bluetooth device for later

use in further communications instead of performing pairing at every attempt to communicate with the Bluetooth/Demo board device.

This attack on the Bluetooth PIN code is estimated to be high if the pairing process of the Bluetooth/Demo board device with the SSO MIDlet is performed very often. Otherwise, the attack based on cracking the PIN code of the Bluetooth device is estimated to be medium due to the fact that the attacker needs to be located in the range of the Bluetooth device (from 10 to 100 meters) when the paring process is started between the SSO MIDlet and the Bluetooth/Demo board device.

The (RMS) storage of the mobile phone is vulnerable and needs to be protected because it is protected only by a master password that the user provides to enter the SSO MIDlet. This authentication method used in this protection mechanism is not secure. It is based on implementing a simple string matching function for comparing the prompted password to the original one. An attacker could easily crack the password and get access to these access credentials that belong to the user of the prototype. Because these access credentials are stored in a plain text without any protection.

The channel between the Demo board and the PC is vulnerable and needs to be protected because the PC, which the Demo board is connected to, is not protected. An attacker could install a Trojan horse in order to get information that may help in retrieving the access credentials or getting directly these access credentials as described in section 3.2.1 of this chapter.

4. Enhancement of the SSO System

4.1 Design Requirements of SSO System

The SSO prototype is composed from three main devices the PICDEM FS USB Demo board, the SonyEricsson w550i mobile phone and the F2M01 Bluetooth devices. During the design of the security protocol, we were always thinking about how to come up with a practical and secure design that overcomes the limitations, which were resulting from the weak processing and memory power and from the technical restrictions of the developing platforms of the Demo board and the mobile phone devices. In this section we are going to discuss these limitations

4.1.1 The PICDEM FS USB Demo Board Device

This device acts as a keyboard emulator in the prototype, full description is provided in section 3.1.3.1. The Demo board is manufactured by the microchip company. It belongs to the PIC18F4550 category of microchip, which can perform only 8-bits processing power and 2 K byte of RAM (1K byte Dual Port RAM + 1K byte GP RAM)[25]. The most complicated technical problems we experienced during our work on this thesis for implementing secure cryptographic algorithms in the security protocol were due to:

- The limited processing power and memory size of this device.
- The limited performance of its software package in supporting and developing cryptographic algorithms.

These two limitations caused many technical complications that were difficult in some cases to overcome.

4.1.1.1 Hardware Limitations

The Demo board is a device with 2 K byte RAM size, which is composed from 1Kbyte of Dual Port Ram and 1 K byte of GB Ram. The size of this RAM is considered to be a very small for most of the symmetric and asymmetric cryptographic algorithms. Most of these algorithms recommend 32 k byte RAM size for implementation, especially the standard one that considered being secure by the cryptanalyst community.

The processor power of this device is very limited too. It has only an 8-bits processor power, this allows only the implementation of a few light weight cryptographic algorithms like the RC4. Most of the 8-bits cryptographic and hashing algorithms e.g. AES-128 and SHA-1could not be implemented in the Demo board because of its low processing power. The memory size could not be scarified only to the encryption and decryption operations. Even in the case where we scarify most of the processing power of the Demo board to drive the AES-128 and risk the SSO system to operate in a very slow mode, we will still have to deal with the limitation of the memory size of this device, which is difficult to overcome. The same problem implies for the SHA-1, we thought that we could run the SHA-1 in the Demo board. But later we discarded this plan and choused to don't take the risk of running the prototype in a slow mode or make it crash from time to time.

4.1.1.2 Software Limitations

The main program used for programming the Demo board device is the MPLAB IDE v7.30 and the C18 compiler v3.02. The MPLAB IDE is a tool for editing assembly codes. The C18 compiler should be integrated to the MPLAB IDE to make it support the ANSI C [27]. The C18 is a version of C dedicated to be used for programming the microchip devices that belong to the micro controllers of PIC18 category. The main limitation of the MPLAB IDE/C18 is that it is library functions don't support any of the cryptographic algorithms needed for the security protocol.

The MPLAB C30 is the latest version of this software [41]. It was developed to program the microchip and controller used in programming the ATM machines but it could be used too in other environments such:

- Point of Sale Terminals that communicates with servers.
- Intruder Alert Systems, which is installed in houses, buildings, cars and communicate with security agency.
- Wireless and hand held devices that interact with secure services or other access controlled devices.

The dsPIC30F cryptographic library that belongs to this version is composed from two software packages [40].

- The dsPIC30F Symmetric Key Embedded Encryption Library, which offers implementations of the Advanced Encryption Standard (AES), Triple Data Encryption Algorithm (Triple DES), Secure Hash Algorithm (SHA-1) Message Digest Algorithm (MD5) and the ANSI X9.82 deterministic Random Bit Generator algorithm.
- The dsPIC30F Asymmetric Key Embedded Encryption Library, which offers the implementations of the Rivest Shamir Adelman (RSA), Digital Signature (DSA) and Diffie Hellman Key Agreement Protocol.

We thought that we can use this version and benefit from its library functions. However, the C30 is developed for programming microchip of 16 and 32-bits power. Another issue is that the dsPIC30F cryptographic package could not be used in developing a secure protocol for the prototype because this package is recommended to be used only with powerful microcontroller than the Demo board device. It is recommended only for 16 and 32-bits process power too [40].

These were the major technical limitations that we experienced when we were investigating the technical specification of this device. We tried during our work in this thesis to integrate an 8-bits SHA-1 C implementation and 8-bits AES C implementation to the design of the Demo board. We successed to get a free fail implementation for the SHA-1, but we were not able to burn the compiled code into the fixed memory of the Demo board device. Even microchip devices that belong to a powerful category than the Demo board device as the dsPIC30F6010/6011/6012/6013/6014 devices are not able to implement the Big Integer Arithmetic Package and consequently the Asymmetric key Embedded Encryption library [40].

4.1.2 The SonyEricsson W550i Mobile Phone

The limitation for this device is almost in the software platform used for developing the SSO MIDlet. The Standard Java environment (J2SE) offers the Java Cryptograph Architecture (JCA) and the Java Cryptograph Extension (JCE) for securing the application developed for the mobile phone environment. These packages offer a lot of cryptographic services as implementation of the symmetric and asymmetric algorithms, digital signatures, key generators and message digest [42]. However, the CLDC/MIDP environment used in developing the SSO MIDLet doesn't support these packages [43]. In order to integrate some security measures to the SSO MIDlet, the algorithms should be self implemented, which is not an appropriate task in this stage due to the restricted time for this thesis. Usually these cryptographic algorithms are implemented as a special version to fit the technical specification of the mobile phone e.g. processing power and memory size.

However, this is not a big limitation because there are other cryptographic packages like the Java Bouncy Castle [45], IAIK JCE Micro Edition [46], Phaos Micro Foundation [47] and NTRU Neo for Java Package [44], which offer Java implementation for the most standard known symmetric and asymmetric algorithms, the key generation, elliptic curve cryptograph and message authentication. In this thesis, we are going to use the Bouncy Castle Package. This is due to the fact that the previous work that was related to the SSO system during previous thesis made use of this package [51]. This package is considered to be a light weight API, which has a similar structure to the JCE (Java Cryptography Extension). The Java Bouncy Castle Package could be integrated into the J2ME development environment without any technical problems [45]. During this thesis we studied the implementation of some cryptographic algorithms that we thought we could make use of in the security protocol. We found out that the only limitation for the Java Bouncy Castle Package is that the cryptographic algorithms are implemented in their minimal version as the AES-128 or unsecured implementation as SHA-1 that was implemented without the padding function.

The limitation related with the algorithm implementations discussed above are due to the fact that the hardware of the SonyEricsson w550i mobile phone device has limited processing power, limited memory and low bandwidth. These hardware limitations don't support long encryption keys or elliptic curves that are known to give better security because it requires from the mobile device more computing power, which would result in scarifying more computing time to perform the task [43] [49].

4.1.3 The F2M01 Bluetooth

This device has no technical limitations. It acts as bridge between the mobile phone and the Demo board devices. It has broadcasting range from 10 up to 100 meter. The only thing that should be done is to configure it for the first time to a PC in order to make the pairing process with the mobile phone device. This is not a limitation but a normal technical procedure.

4.1.4 Securing the Communication

There are three channels that the security protocol should protect in the prototype:

- The channel between the mobile phone and the Bluetooth/ PICDEM FS USB Demo board device.
- The channel between the user of the prototype and the mobile phone where all the access credentials are stored in the (RMS) of the mobile phone.
- The channel between the Demo board device and the PC.

The main channel that should be protected is the first one. The access credentials are sent from the SSO MIDlet through the Bluetooth radio-broadcasting waves in a clear text without any kind of protection to the Demo board. This considered as a very a critical situation because it makes the prototype very vulnerable and exposes its users to number of serious security threats. However, securing this channel turns to be a quite complicated task because of the limited processing power and memory size of the Demo board device that acts as key board emulator in this prototype. The access credentials sent from the mobile phone to the Demo board devices should be encrypted with a secure algorithm to make this channel secure enough. However, it was quite challenging task to find a secure cryptographic algorithm that:

- Fits the limited processing power and memory size of the Demo board device
- Fits in minor grade the limited processing power and memory size of the mobile phone device.
- Could be compiled and integrated into both developing environment, the MPLAB IDE/C18 and the Netbeans IDE.

Omitting one of the previous requirement will result on developing a security protocol that is secure enough but not practical because it could not be implement properly in the design of the prototype.

The bottle link we faced during the design of the security protocol was the Demo board device. Most of the standard secure algorithm as the AES and the SHA-1 could not be implemented because of the limitations discussed above. For the mobile device we could implement both symmetric and asymmetric crypto algorithms but only in 8-bits mode with a small size key and some recommended tuning to the algorithm to make it fit the limited processing and memory size of the mobile phone. This issue raised many concerns questions about the performance and the security of these implementations.

The second channel that the security protocol should secure is the channel between the user and the mobile phone. The access credentials are stored in the (RMS) of the mobile phone device in a clear text format. If the SSO MIDlet is not well protected, an attacker could easily get access to the mobile phone and then to the sensitive data, which is composed from the access credentials and the master password. In the prototype, this sensitive data are stored in a clear text without any kind of protection. The security protocol should:

- Assure a good authentication mechanism of the user to the SSO MIDlet.
- Make sure that the access credentials, which are stored in the (RMS) of the mobile phone are well protected through encrypting them with a secure encryption algorithm.

The third channel that should be protected is located between the Demo board device and the PC. The demo board device is connected to a PC with a standard USB cable. This channel is vulnerable to some serious attacks as we described in section 3.2.1 of this paper. However, protecting this channel is quite complicated task. If we are going to include securing this channel as a requirement for the security protocol we will not comply with the mobility requirement of the SSO prototype. This due to the fact that if we are going to secure this channel we will be obliged to install a software into the PC, which the Demo board device will be connected to. By doing so, the user will be forced to use only some specific PC or networks.

From the previous discussion above we conclude that protecting the first and the second channel represent major security requirements that should be taken in consideration during the design of the security protocol. However, a solution is needed for protecting the third channel.

4.2 System Design Methodology

In order to make the security protocol meets the security requirements of the SSO system. Realistic and secure solutions are needed to fit with the different limitations resulting from the limited processing power, memory size and developing platforms related to the devices used by the prototype. In the same time these technical solutions should make the prototype secure against the different attacks that is vulnerable for. However, the experience learned from previous systems proved that no computer system is fully secure. The security is measured by percentage but a certain security benchmark should be achieved if the prototype is going to be commercialized for the future. The security protocol who could offer the best security levels to the prototype is the protocol who can achieve a good compromise for implementing secure cryptographic algorithms into the limited processing power and small size of the memory of the Demo board and the mobile phone device.

4.2.1 Securing the Bluetooth Communication

The best way to secure the channel between the Bluetooth/Demo board and the mobile phone devices is by encrypting it with the RC4 symmetric cipher algorithm. The RC4 is used very frequently for systems similar to the SSO prototype, which has limited processing power and storage size [21]. This is due to that the RC4 is very speedy and simple algorithm. The RC4 is still the algorithm of first choice for the applications developed for ATM and credit cards applications, which have matching hardware power as the Demo board device.

Even with the recommendation from the cryptanalyst community that mentions that the RC4 is not longer considered to be a secure cryptographic algorithm, the RC4 is still implemented by many common security protocols e.g.:

- Secure Sockets Layer (SSL) for protecting the internet traffic.
- WEP for protecting the wireless networks.
- WPA for protecting the wireless card environment.
- TLS.

The RC4 is implemented too in many other wind range applications this due to its impressive speed. It is very simply implemented in developing both software and hardware systems.

The RC4 is a symmetric cipher algorithm, which is implemented through a pseudo random bit generator as mentioned in section 2.1.2. The output of this step should be XORed into the plain text or cipher text that is worked on. The RC4 algorithm can operate with variable lengths up to 2048. In order to secure the communication going through this channel, we have to ensure the correctness of the RC4 implementation. The RC4 could be insecure algorithm if the encryption key is not linked to the initialization vector. This would result in the possibility of having two plain texts, which are encrypted with the same output stream. The encryption key in the security protocol we are developing for the prototype should be generated in a very secure way. Because if we secure this key we can be sure about the security of the basic scheme in the protocol but if the key is generated in an insecure way we could make the encryption scheme of the protocol very vulnerable because a previous attack in the RC4 proved the possibility of computing this key from one or four million plaintext- cipher text pairs [02].

4.2.2 Securing the RMS

This channel exists between the user and the SSO MIDlet installed in the mobile phone device. In order to secure this channel the security protocol should provide:

- Good authentication mechanism, which authenticates the user to the SSO MIDlet.
- Secure encryption algorithm, which protects the access credentials stored in the (RMS) of the mobile phone device.

These two requirements need to be met for securing this channel.

4.2.2.1 The Authentication Mechanism

The security protocol should provide the SSO prototype with a secure authentication mechanism. This is considered as fundamental requirement for the protocol. In order to provide this authentication mechanism, we have to implement a secure hashing algorithm that could fit the processing power and the memory size of the mobile device. The Java Bouncy Castle Package along with other package mentioned in section 4.1.2 of this paper offer many hashing algorithm that are excitable in quite reasonable period of time by the mobile phone device.

As mentioned in section 4.1.2 of this paper, the CLDC/MIDP doesn't support the Java Cryptograph Architecture (JCA) and the Java Cryptograph Extension (JCE). The Java Bouncy Castle Package will be used in this thesis because is made free and used in previous thesis related with mobile phone security.

The SHA-1 hashing algorithm will be used during the authentication process of the user to the SSO MIDlet. One of the major recommendations for the prototype is to use only one master password. The security protocol will use this master password for authenticating the user to the SSO MIDlet. However, due to the design restriction of the security protocol, the authentication mechanism will make use only of the first 32-bits of the hashed value of the master password. The hashed value of the master password will be stored in the (RMS) of the mobile phone and used during the authentication procedure and the keys generation later by the protocol. The user should provide its password to the authentication mechanism, which will generate its hashed value and compare the first 32-bits of this value to first 32-bits of hashed value of the master password, which is stored in the (RMS) of the mobile phone. If the both values are equal then the user will be successfully authenticated to SSO MIDlet. If not the user will be denied from getting access to the SSO MIDlet and the authentication process will terminate.

This authentication process offers a good method of authentication because it enhances the over all security of the authentication mechanism by making use of the hashed value of the master password. The hashing methods avoid any possibility of tempering with the original value of the master password. The user during this process will need only to provide its master password to the SSO MIDlet.

The choice of using the SHA-1 as an authentication method is because it is considered as the safest hashing algorithm provided by the Java Bouncy Castle Package implementation. The SHA-1 is implemented too in many other popular and widely used security protocols as the TLS, SSL, PGP, SSH, S/MIME, and IPsec. However, this version of the SHA-1 is an 8-bits implementation without padding. It is executed with time average of 667 (ms) in the k-500 SonyErikson mobile phone, which has a similar processing power and memory size as the mobile phone device we are using in this prototype [51].

4.2.2.2 The Encryption Mechanism for the RMS

In order to secure the access credentials stored in the (RMS) of the mobile phone, the security protocol should implement a secure encryption algorithm. This considered as a fundamental requirement that the security protocol must meet. The same factors faced for the task discussed in section 4.2.2.1 apply to this task too. When we were looking for a good method for securing the access credentials stored in the (RMS), we had to take in consideration the limitation of both the mobile phone device and its Java developing environment described in section 4.1.2. In order to solve this issue, we decided to make use of the 8-bits block cipher Advanced Encryption Standard (AES) implementation provided by the Java Bouncy Castle Package. This AES-128 implementation is considered to be secure enough to protect the access credentials, which are stored in the (RMS) of the mobile phone device as mentioned by the NISIT in their assessment report for the algorithm [02]. It is better to use the AES with key length of 192 or 256 but again the limited processing power and the storage memory size of the mobile phone recommended the deployment of the AES 128key length.

Another consideration is that usually the AES is implemented together with the Secure Hash Algorithm (SAH-1). This combination proved to be very good. Many protocols use this combination for many systems with different technical background.

The user of the prototype needs only to provide his master password in order to authenticate himself to the SSO MIDlet. The same master password is used for encrypting and decrypting the access credential of each account that belongs to the user and stored in the (RMS). The master password and the name of the account will be used to generate an encryption key that will be used for the encryption and decryption process of every access credential of every account. This encryption key should remain discreet for the user of the prototype.

The time used for executing the AES-128 algorithm implemented by the Java Bouncy Castle Package in K500 SonyEricsson mobile phone, which has almost the same processing power and memory storage size as the mobile phone used in this prototype is 601 (ms) [51]. This is considered to be quite a reasonable time if we take on consideration the limitation of the mobile device and the nature of the SSO MIDlet application.

5. Detailed Description of the SSO Security Protocol

In order to secure the SSO prototype, we have to protect:

- The Bluetooth communication between the Bluetooth/ PICDEM FS USB Demo board and the mobile phone devices.
- The sensitive data stored in the (RMS) of the mobile phone device.
- The communication between the Demo board device and the PC.

The prototype is vulnerable because these three parts remain without good protection. An attacker could perform many attacks introduced in section 3.2.1 of this paper. However, due to the recommendation for the prototype to be a mobile system, the security protocol we are designing during this thesis will focus only on securing the first and the second part. The third part could not be protected without scarifying the total mobility of the prototype and restricted it only to a specific PC or networks. This solution, even if it doesn't meet the requirement of the SSO system for this thesis, it still represents a good solution for private environment where the user of the SSO system usually logs in to a specific PC or network e.g. PC in his office or home, which is easy to be targeted from an attacker, especially if the PC is located in an open environment or if it is a laptop.

5.1 Notation

Here we describe the notion used within the protocol.

- (mpw) is a master password.
- (Ek2) is a long-term secret key.
- (k1) is a short-term secret key.
- (k2) is a secret encryption key.
- H is a one way hash function.
- (A) is the account name.
- (AC) is the access credential that is related with every account.
- (P1) is the mobile phone principal.
- (P2) is the Bluetooth/ PICDEM FS USB Demo board device principal.

5.2 Description of How to Protect the RMS

In order to secure the (RMS) of the mobile device, the protocol should implement two security mechanisms:

- An authentication mechanism that authenticates the user to the SSO MIDlet, which is installed on the mobile phone device.
- An encryption mechanism that protect the access credentials, which are stored in the (RMS) of the mobile phone device

5.2.1 Description of the Authentication Mechanism

In order to protect the (RMS), we decided to implement a strong and secure authentication mechanism to authenticate the user to the SSO MIDlet before he will be granted any kind of access to the prototype.

During the authentication process the user will be asked to authenticate himself to the SSO MIDlet. According to the fundamental requirement of the SSO system, which recommend only one password for every user of the prototype, the security protocol must use only a master password for different purpose. The user must have only one password to perform the entire log in tasks. In order to comply with this recommendation we decided to implement the SHA-1 hashing algorithm to check for the correctness of the master password.

The authentication mechanism is based on hashing the master password of the user, Auth = H(mpw2). Only the first 32-bits of the master password hashed value will be used for the authentication purpose. The H(mpw1) = H(mpw2) where the (mpw1) = (mpw2) = master password. The SSO user should possess the mpw2 while the H(mpw1) should be stored in the (RMS) of the mobile phone. This first 32-bits of the H(mpw2) will be compared to the first 32-bits of the H(mpw1). The H(mpw2) should be generated every time the user needs to log in to the SSO MIDlet.

Now we list the different steps of the authentication mechanism implemented by the protocol:

- First step, the user must provide a master password to the SSO MIDlet.
- Second step, the security protocol will hash the master password.
- Third step, the authentication mechanism should compare the first 32-bits of the generated hashed value form the previous step to the first 32-bits of the hashed value of the master password, which is stored in the (RMS) of the mobile phone. If they are identical, then the authentication procedure was successful and the user will be granted access to the SSO MIDlet, if not the access will be denied and the authentication procedure will terminate.

This mechanism is quite simple, secure and efficient. By implementing it into our protocol, we will be able to meet the requirement of that a user must possess only a single master password for getting access to every service offered by the SSO system. The use of only 32-bits for the authentication purpose from the hashed value of the master password is related to the design recommendation for the long-term secret key generation introduced later in this paper. This mechanism should be integrated to the prototype without any technical complication because the mechanism is based on hashing algorithm that could be implemented by the mobile phone hardware processing power.

5.2.2 Description of the Security Mechanism for the RMS

All the accounts, which are created by the user of the SSO MIDlet have its related access credentials that are stored in the (RMS) of the mobile phone devices. These access credentials are considered to be highly sensitive data hence the security protocol should protect them very well. In the prototype, these access credentials are stored in the (RMS) as clear text. The only protection that was offered by the prototype to protect them was a simple authentication function. This situation is considered to be very critical from a security point of view since getting access

to these access credentials is a very easy task. Many possible attacks could be conducted against the mobile phone devices to retrieve these data as described in section 3.2.1 of this paper.

In order to protect these access credentials, we decided to implement a security mechanism for the (RMS). However, it is important to mention that the authentication mechanism described in section 5.2.1 is considered as an independent mechanism from the protocol but this security mechanism is considered as a part of the security protocol. This mechanism is going to be implemented by the mobile phone device. This factor will allow us to jump over the complication resulting from the limitation of the computation power of the hardware devices of the prototype. We decided to make use the AES-128 implemented by the Java Bouncy Castle Package.

After the SSO user successfully authenticates himself into the SSO MIDlet, he will be able to get access to all the accounts stored in the (RMS) of the mobile phone. The security mechanism should be executed automatically when the SSO user is attempting to access these accounts. However, it is worth to mention that the decryption method will be executed first during this process because in the protocol all the accounts are supposed to be saved encrypted in a cipher text format in the (RMS) of the mobile phone. The encryption will be executed after the user finishes using the account and is going to shift for using another account, when he is going to log off from the SSO MIDlet or when he is going to log in to that account. The only time when the encryption is performed first is when the user creates a new account. The overall design of the (RMS) security mechanism is shown in figure 2.

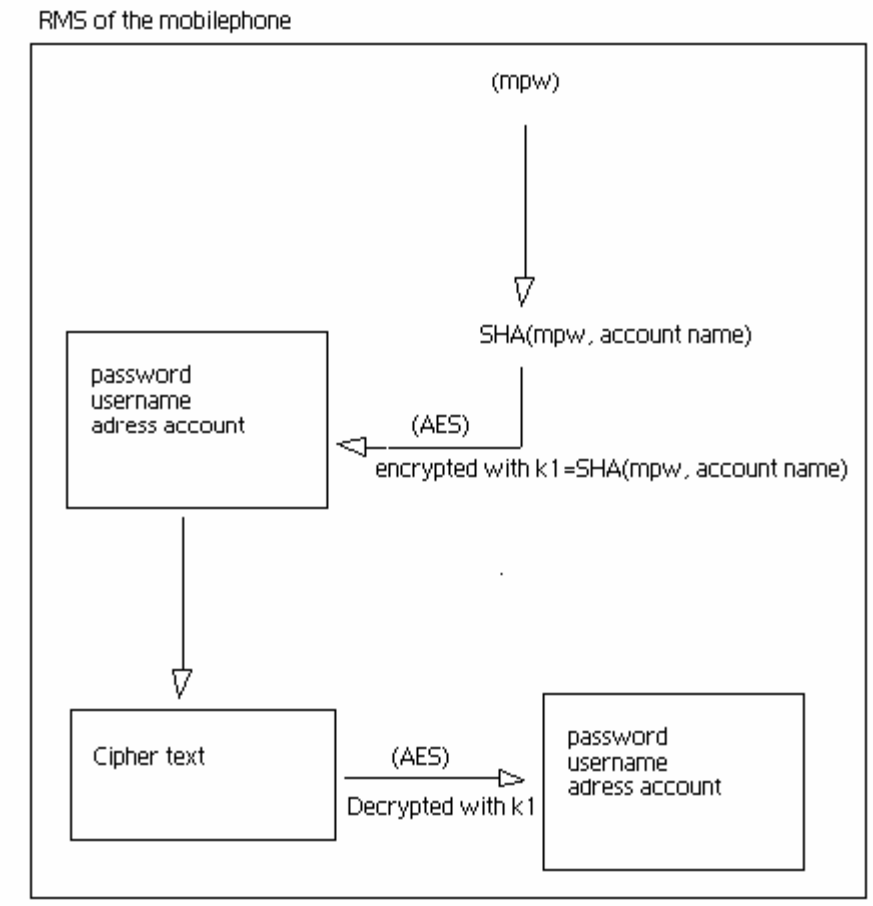

Figure 2: shows a Scheme of the RMS Security Mechanism

The security mechanism is composed from two major steps:

- The generation of the encryption key by hashing the value of the master password by using the SHA-1 Java Bouncy Castle implementation and then XORed it with the value of the account name.
- The encryption and decryption of the access credentials by using the AES-128 Java Bouncy Castle implementation with the encryption key generated from the previous step.

Every account should be encrypted and decrypted with its own short-term secret encryption key (k1). The mechanism generates a short-term secret unique key (k1) for every account because this key is generated by first hashing the master password and XORed its hashed value with the account name. By using the account name in the generation process of the key (k1) we will be able to produce a unique short-term key (k1) for every account. But we have to make sure that the account names, which are created by the user and stored in the (RMS) of the mobile phone are different from each other and unique. By doing so, we will simplify the generation process of the short-term secret keys (k1) for every account. The generated secret keys should be of 128-bits length as specified above. Longer key gives better security level. However, we are always limited by the weak computation power of the mobile phone device. The AES is considered to be a secure cryptographic algorithm even with a key length of 128-bits.

The user doesn't need to know the value of the short-term secret keys (k1) used for the encryption and decryption of the access credentials by the AES cryptographic algorithm. It remains discreet so far the master password is not compromised. Every account should have its unique encryption key (k1). This key is considered to be a short-term secret key, it will be generated every time when the user needs

- To access an account in the SSO MIDlet
- To move to another account in the SSO MIDlet.

For achieving better security, the security mechanism should avoid saving this short secret encryption keys (k1) that belong to the accounts saved in the (RMS) of the mobile phone device. It is better and more secure to generate them only when the user is going to access its relative accounts. An attacker could get access to these keys if he will get access to the mobile phone device and forward to its (RMS). But after integrating this mechanism to the system, the attacker must

- Get access to the mobile phone.
- Compromises the master password and get access to the account names.
- Computes the value of the secret key.

However, this is not an easy task because the attacker will need to compromise the master password (mpw).

By implementing the authentication and security mechanism described above in this section, the security protocol will enhance the overall security of the prototype by eliminating most of the attacks dedicated against the (RMS) of the mobile phone device as described in section 3.2.1 of this paper. The protocol will limit the access to the SSO MIDlet only to its legitimate user and secure the access credentials by encrypting them and keep them confidential.

5.3 Description of the Core Security Protocol

The main task of designing the security protocol during this thesis for the SSO prototype is to secure the Bluetooth channel between the Demo board and the mobile phone device. Through this channel the access credentials are broadcasted in clear text. This exposed the entire prototype to many threats that made it very vulnerable. An attacker doesn't need very sophisticated tools or deep knowledge for conducting attacks against this communication channel and retrieve these access credentials. Many possible attacks dedicated against this channel are described earlier in section 3.2.1 of this paper.

However, securing this channel turned to be a complicated task due to the limitation of the Demo board device. This device supposed to act as a keyboard emulator in the prototype. As described in section 3.1.3.1 the limitation related to this device are both hardware and software limitations. This resulted in experiencing many technical problems that we should seriously take in our consideration when we are going to design and implement the security protocol for the prototype. The protocol should:

- Offer a high security level to the prototype.
- Be easily integrated to the prototype without making any major structural changes to its design, which was developed early in a previous thesis.
- Keep the prototype stable, this means that implementing the protocol should not result in overloading the system and cause it to crash.

These are the three major requirements that the protocol should meet to enhance the security of the prototype. In order to achieve a high level of security for the prototype, we have to implement a secure and powerful cryptographic algorithm to secure the Bluetooth radio-broadcasting channel that exists between the Demo board and the mobile phone device. This looks like an easy task because there are many good algorithms that could perform this task. However, because the Demo board device was acting as an achilles heel in this project. We were running out from most of the best options. The cryptographic algorithm that should be implemented to secure this channel should be fast, easy to compute and secure. Perhaps, there are not many algorithms that have all these three characters at the same time. Most of the candidate algorithms we were taking in consideration were secure, fast but need more computing power or more memory size than the Demo board offers. The only algorithm that we believe that will perform well and satisfy the three requirements mentioned above is the RC4 stream cipher algorithm. From the previous experience with big systems e.g. Secure Sockets Layer (SSL), WEP (to secure wireless networks), wireless cards and TLS [04]. The RC4 proved to be light weight fast algorithm, which is secure if it is implemented correctly, here we mean that its secure initial vector (IV) and long encryption key. Full explanation of this process is described in section 2.1.2 of this paper. By implementing the RC4 for securing this Bluetooth channel, we will be able to fulfill the first requirement mentioned above, which is securing the SSO prototype.

The RC4 is a light stream cipher algorithm, and its code implementation is simple. So it could be easily implemented in the MPLAB IDE/C18 platform used to program the PICDEM FS USB Demo board device. The Java Bouncy Castle Package offers an RC4 implementation too but it is not made free. The fact that the RC4 doesn't recommend a big C library to be implemented should be considered as a big advantage to this algorithm because the C library in the MPLAB IDE base C18 is very limited and restricted only to the functions related directly with the embedded environment. Because the RC4 fulfills the first requirement and is itself easy to be implemented so integrating it to the design of the prototype will be a good choice.

We proved this during this work in this master thesis, we was able to compile the RC4 with the MPLAB IDE/C18, integrating it to the software design of the Demo board used for the SSO prototype and burn it to its memory storage.

By meeting the requirements listed above, we will be able to keep the system stable and well performing because the first requirement ensure the security of the system that will enhance its performance, the second requirement will keep the same level functionality for the SSO system and the third requirement ensure its stability because it avoid problems e.g. performing tasks very slowly or crashing due to the overloading of its memory storage and processing power.

5.3.1 How the Security Protocol Work

After the user authenticates himself to the SSO MIDlet, he will be able to access a specific account from the different accounts stored to in the (RMS) of the mobile phone device. During this process a short-term secret encryption key (k1) will be generated. Every account is stored in the (RMS) as encrypted cipher text with the AES. The security protocol should perform these two major tasks. First, encrypt the Bluetooth channel between the Demo board and the mobile phone device and second, log in the user to its account. However, before we introduce the protocol we should mention that the long-term secret encryption key (Ek2) and (k2) should be pre installed in both devices. The (k2) will be generated randomly first and only once in the beginning of the process. The (k2) will be stored in the fix memory of the Demo board device. The (Ek2) will be generated after the (k2). The (Ek2) is generated out from the last 128-bits of the hashed value of the master password XORed with the value of the (k2). The (Ek2) should be stored in the (RMS) of the mobile phone. The overall design of the security protocol is shown in figure 3.

The protocol is composed from these five successive steps as follows:

- Decrypt the access credentials that belong to every account buy using the short-term secret encryption key (k1).
- Encrypt the plain text of this access credentials with the RC4 stream cipher algorithm. In order to perform this task, we should use the long-term secret encryption key (Ek2).
- The SSO MIDlet should send the cipher text through the Bluetooth radio-broadcasting channel to the Demo board device.
- The Demo board device should receive the access credential, which is in a cipher text and decrypt it by using the RC4 decryption method by using the long-term secret encryption key (k2). The (k2) should be installed in the memory of the demo board device.
- After retrieving the plain text of the access credential, the Demo board device, which acts as a keyboard emulator forward the data to the PC, which the Demo board is cobbled to.
- The SSO prototype user should access the targeted account.

The first step should be performed because the access credentials reside in the (RMS) of the mobile phone device encrypted with the AES-128 Java Bouncy Castle implementation. This step is mandatory because the Demo board does not implement the AES algorithm.

The second step is where the data is encrypted and sent through the Bluetooth radio channel to the Demo board device. The most important issue for this step is to use a secure encryption key (Ek2) for encrypting the access credentials. The key (Ek2) should be long enough to be considered as a secure key for the RC4.

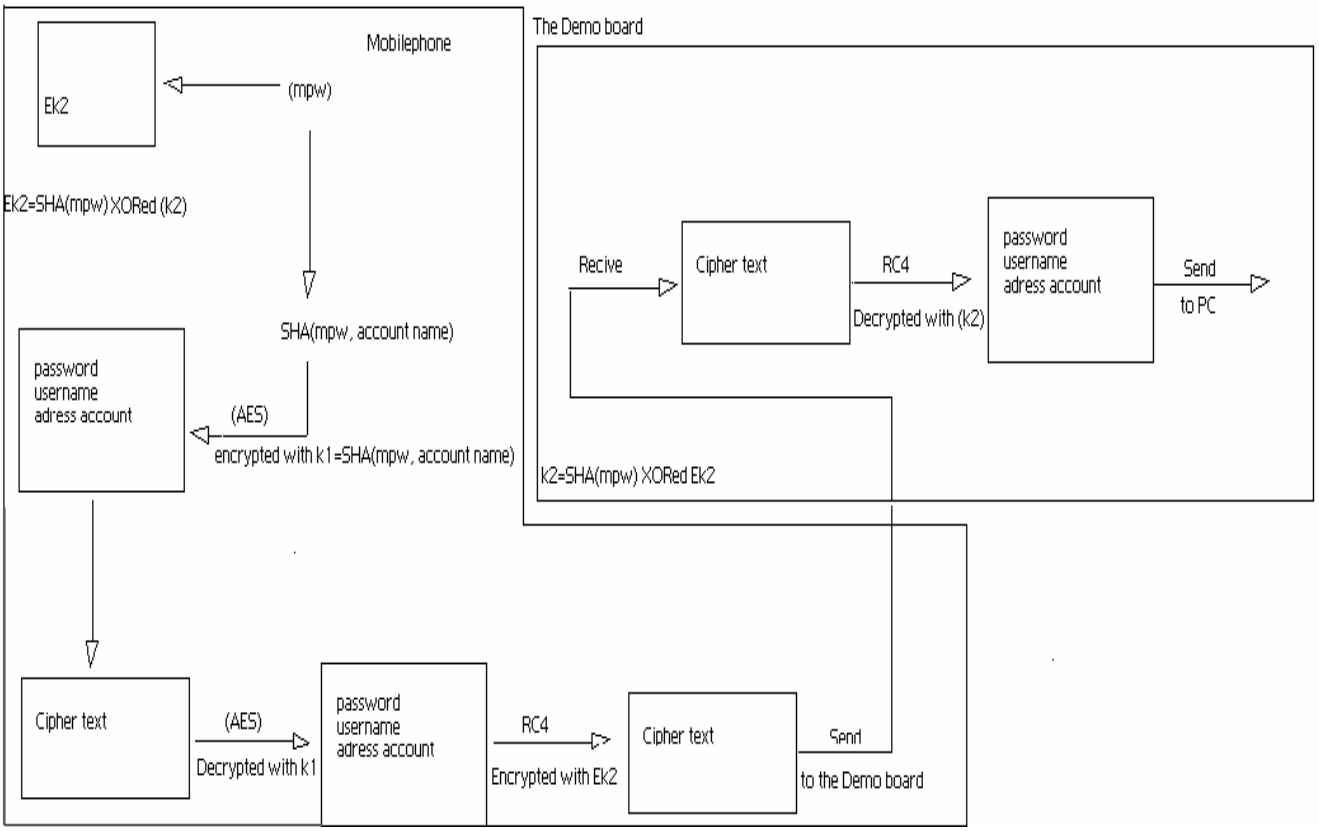

Figure 3: shows the Security Protocol

During the remaining three steps of the protocol the encrypted data will be sent to the Demo board device, when they will be decrypted in the fourth step buy using the secret key (k2) and forward it to the PC where the Demo board is cobbled to. In the PC this access credentials will be used to log in the user to its account. However, for better understanding of these steps involved for securing this channel, we should describe the process the Demo board will perform as a key board emulator in the prototype after the decryption operation is performed in the fourth step. After the enumeration process, the Demo board will set up the USTART interface [52]. This interface is the one that should communicate with the RS232 serial COM port. The Bluetooth is transparent to the PICDEM FS USB Demo board due to the fact that the data is forwarded to this device from the Bluetooth over the RS232 interface, which is considered by the Demo board as a normal serial communication. This means that this demo board is not able to distinguish between the emulated serial traffic and normal serial traffic that is trafficking through the standard RS232 cable. This is an advantage for us to send the sensitive data in a RC4 ciphered text during the second steps and the third step of the security protocol mentioned above. In order to perform as a keyboard emulator, the Demo board registers an interrupt function, which listen to the RX receive pin of the RS232 interface. For every time the RX pin receives a byte, the demo board should evoke the interrupt function, which is responsible for reading and storing the new received byte in to the global circular buffer of the demo board device. Because the SSO MIDlet sends only one

byte at a time, we should wait for decrypting the RC4 ciphered sensitive data until the ring buffer of the Demo board contains full 8 bytes or more. However, the decryption should be implemented before another function will read the 8 bytes from the buffer and place them into the input report. Because the input report is the final step of data processing, after the data is placed in the inputs report, it will be sent directly to the PC the demo board device is connected to. The security protocol should make sure that the data placed in the input report is in plain text format or the user of the SSO prototype will not be able to log in to its account. The decryption process should be implemented iteratively until the buffer will be empty and all the characters are displayed in the screen of the PC. This is the description of the major technical issues that the design of the security protocol should take into consideration.

6. Security Analysis

6.1 The Protocol

There are two main entities involved in this protocol, the mobile phone and the Bluetooth/ PICDEM FS USB Demo board device. The mobile phone device performs two key computations. (The execution of the security protocol is shown in figure 4).

- The long-term secret encryption key (Ek2) is used at a later for the encryption of the RC4 stream cipher data. The (Ek2) computation is based on XORing the last 128-bits of H(mpw) with the (k2). The (k2) is a long-term secret key randomly generated first and only once in the beginning of the process. The (Ek2) is stored in the (RMS) of the mobile phone and remains discreet even for the legitimate user of the prototype.
- The short-term encryption keys (k1) used to decrypt the access credentials that belong to an account residing in the (RMS) of the phone in a cipher text format. This cipher text is encrypted earlier by using the 8-bits AES Java Bouncy castle implementation by the user of the SSO MIDlet. The (k1) keys are used for the encryption and decryption of the access credentials that belong to the accounts (A). Every account has its own short-term secret encryption key (k1), which is generated by hashing the value of the master password (mpw) with the account name (A), k1 = H(mpw, A)

These are the two key computations that must be performed to make the protocol work properly. The two keys are computed by the mobile phone principal (P1) and stored in its (RMS).

In order to communicate with each other, (P1) and (P2) must share the same key. The (Ek2) and (k2) are replicas from each other's. The (Ek2) = the last 128-bits of H(mpw) XORed (k2) while the (k2) = the last 128-bits of H(mpw) XORed (Ek2). The (Ek2) is a long-term secret encryption key for the RC4 stream cipher algorithm used by (P1) to encrypt the (AC). The (k2) is a long-term encryption key used by the (P2) for decrypting the received (AC). During the execution of the protocol, (P1) will always use the (Ek2) for encrypting while (P2) uses the (k2) for decrypting the (AC).

During the protocol, execution P(1) performs the following four successive operations:

- Generate the (k1).
- Decrypt the cipher text of the (AC) with the encryption key (k1) for the AES-128 Java Bouncy Castle.
- Encrypt the retrieved plain text of the (AC) with the encryption key (Ek2) and the RC4.
- Send the cipher text to P(2).

At the other end, P(2) responds by performing the three following successive operations:

- Receive the cipher text of the (AC).
- Decrypt the cipher text of (AC) by using the key (k2) for the RC4.
- Send the retrieved plain text of the (AC) to the PC.

The scheme below shows the main successive steps that the two principals performs during the protocol with the correct timing of every key involvement during the protocol

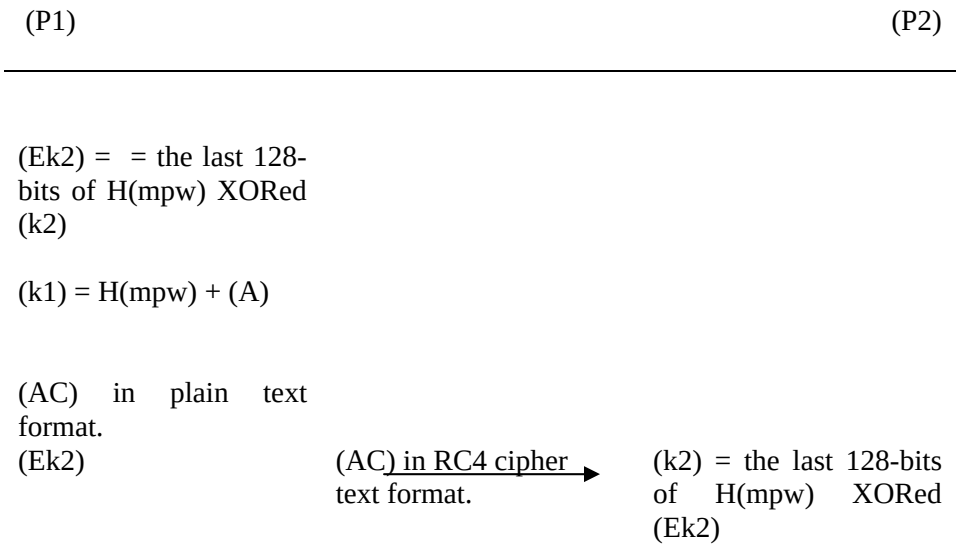

Figure 4: shows an execution of the Security Protocol.

6.2 Analysis of Protocol

The protocol involves two entities: the mobile phone (P1) and the Demo board (P2). In the first step of the protocol scheme, the mobile phone always starts the communication. The Demo board receives the request from the mobile phone and forwards it to third principal, which is the PC. The third principal is omitted here because its role is irrelevant in the protocol. The (P1) must possess the long-term secret key (Ek2) prior to the key generation process of the short-term secret key (k1). The (Ek2) = the last 128-bits of H(mpw) XORed K2. Both the H(mpw) value and the long-term secret key (k2) must be generated before the (Ek2).This is due to the fact that the H(mpw) is involved in the construction of (k1) while the (k2) is involved in the construction of the (Ek2). The (k2) is generated only once in the beginning of the process and is stored in the fix memory of the (P2). In the protocol specification, the (k2) is generated before the (Ek2). The (Ek2) is a replica of the (k2). The (Ek2) is derived from the function (k2) = the last 128-bits of H(mpw) XORed (Ek2). The H(mpw) involved in the generation of the (Ek2) is generated only once in the beginning of the process before the generation of (Ek2). It is stored in the (RMS) of the mobile phone. Only the last 128-bits of the H(mpw) are used for the generation of both long-term secret keys (k2) and (Ek2). The generation of (Ek2) is performed only once by the (P1) in the beginning of the process. Afterwards, the protocol make uses of the stored value of (Ek2) in the (RMS) of the mobile phone.

During the second step of the scheme, the (P1) generates the short-term key (k1) = H(mpw) + (A). Every account must have its own unique short-term secret key (k1). In this process the full length of the H(mpw) is used for the generation of the (k1). The generation of the (k1) is a repeatable process, which is executed every time the user accesses one of its accounts (A). After using the account, the value of the (k1) is discarded and never stored in the (RMS) of the mobile phone. If the user accesses account (A2), (k1) becomes equal to H(mpw) + (A2). The value of the key (k1) is dynamic and different from one account to another unless the two accounts are having the same name. A restriction to impose distinct and different values of (k1) must be added during the implementation process.

The remaining steps in the protocol include decrypting the (AC) with the AES, then encrypting it the (AC) with the RC4 and sending the encrypted value to the (P2), which decrypts it with the (k2). During these steps, there is no key generation or hashing value generation.

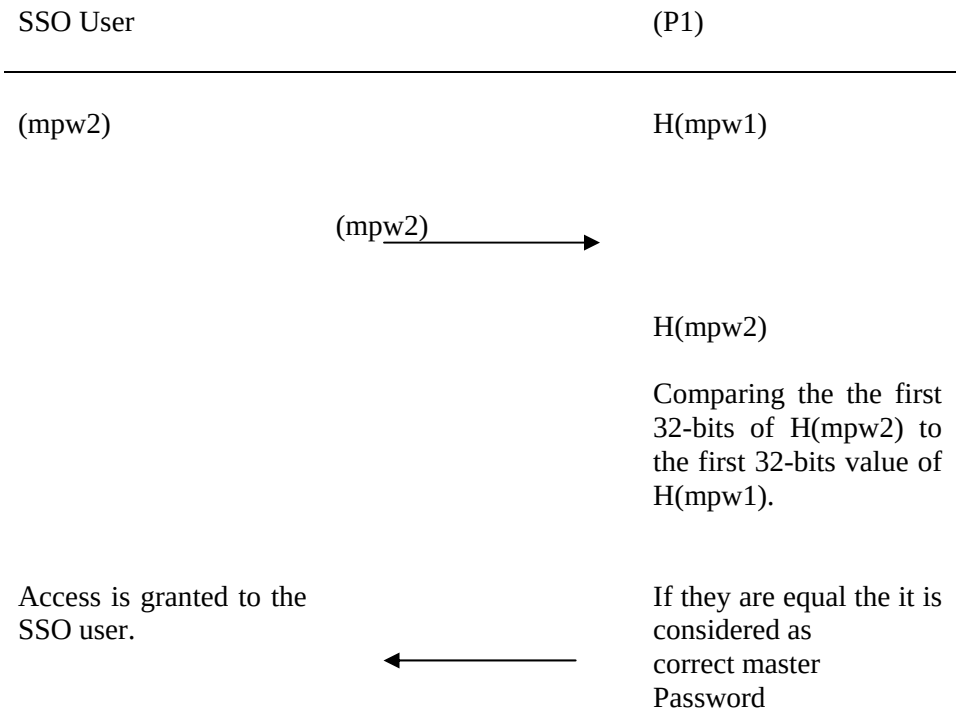

Figure 5: shows an execution of the Authentication Mechanism.

This protocol could be vulnerable to the man in the middle attack. This attack can take place when one of the two long-terms secret keys (Ek2) or (k2) are compromised or when the master password (mpw) is compromised. It is unlikely to compromise the (Ek2) if the master password (mpw) remains secure because the value of the (Ek2) is generated only once and stored in the (RMS) of the mobile phone. The (Ek2) remains discreet even to the user of the mobile phone. Its value can only be revealed by directly accessing the (RMS). However, this is not considered a practical attack in this case. The only possibility for getting access to the (k2) is by stealing (P2), i.e. the Demo board device. This cannot be considered as an actual attack because the man in the middle attack is based on eavesdropping concept, which is not the case here. Stealing the (P2) device cannot take place without the user being aware of it.

The security protocol introduced above for the SSO prototype implements a mechanism that authenticates the user to the SSO MIDlet. The user of the SSO MIDlet has to provide the authentication function with the (mpw). This (mpw) is hashed. The first 32-bits of the hashed value are then compared to the first 32-bits of the hashed value of the (mpw), which is stored in the (RMS) of the mobile phone.

$$\begin{aligned} \text{Auth} &= \text{H(mpw1)}\,(\text{the first } 32\text{-bits of hash value}) \\ &= \text{H(mpw2)}\,(\text{the first } 32\text{-bits of hash value}) \end{aligned}$$

Where H(mpw1) is the one stored fix in the (RMS) of the phone and H(mpw2) is the one provided by the user to the authentication function of the SSO MIDlet. The execution of the authentication mechanism is shown in figure 5.

During the first step of the authentication protocol, the system generates the H(mpw1). This value is generated only once in the beginning of the security protocol process and is stored in the (RMS) of the mobile phone (P1).

During the second step, the user provides a master password (mpw2) to the authentication function. The password (mpw2) should be an exact copy of the (mpw1) performed earlier in the security protocol for the hashing function H. The authentication mechanism will then generate the H(mpw2) and compare the first 32-bits of the hashed value of the (mpw2) to the first 32-bits of the hash value of the (mpw1).

During the third step, the authentication mechanism grants access to the owner of the password if there is a match between the first 32-bits value of both hashed values. Otherwise access is denied.

The hashed value H(mpw1) remains secret and discreet even the owner of the password. The protocol protects the H(mpw1) value by storing it into the (RMS) of the mobile phone, which is access restricted by the master password (mpw). The only way to get access to the (RMS) is by stealing the mobile phone itself, and driving a brute force attack or password guessing attack for getting a possible match with the master password. However, stealing the phone is not considered as a practical attack because the user will discover it. The second option is to try to compromise the master password without stealing the phone but by rather making quick and short tries when the owner is away such as for instance talking with a college in the neighbor office. This is not considered a practical attack because the process of generating master passwords for the system ensures the generation of secure master passwords and avoids the generation of easily guessable passwords.

The authentication protocol uses only the first 32-bits for the authentication purpose. This is due to the design that enforces saving the last 128-bits of the master password hashed value H(mpw) for the generation of the long-term secret keys (k2) and (Ek2). The 32-bits are adequate for providing a secure authentication mechanism.

The protocol also provides some sort of authentication of the two communicating principals (P1) and (P2) against each others. This consists in the fact that the (P(2) is the only entity possessing the (k2), which has only one replica i.e. the (Ek2) stored in the (RMS) of the mobile phone. A user can easily determine whether he/she is communicating with the right demo board device (P2). Because, if he/she sends an (AC) with its (P1) and could not log in into the account successfully, then that means that the mobile phone (P1) is not communicating with the legitimate Demo board device (P2).

Usually complicated protocols provide an authentication mechanism where the communicating principals can verify the identity of each other. There are two authentication mechanisms that are used to perform this kind of a task [53].

- The public key based key agreement protocols.
- The password based key agreement protocols.

The first mechanism is not relevant because of the nature of our protocol. However, the second one can be. The two principals in this case must share a secret password in advance, which will be used for the authentication process. These types of protocols must achieve one of these following goals:

- Implicit Weak Key Confirmation, where (P2) is guaranteed that (P1) is the only other principal that knows the value of the replica long-term secret key of (k2), which is the (Ek2) in the protocol used here.
- Explicit Strong Key Confirmation, where (P2) is guaranteed that (P1) is the only principal that knows the correct key. In other words, (P2) has the possibility to verify whether (P1) has received the correct key.

The protocol achieves the first goal but not the second one. (P2) is guaranteed that (P1) is the only entity in the protocol that knows the value of the replica key of (k2), i.e. is (Ek2) since this key is generated only one time and stored in the (RMS) of the mobile phone. The value of this key remains secret expect for both principal (P1) and (P2), which has the original key (k2).

The goal of the protocol was to secure the Bluetooth communication between the two principals. This goal is clearly achieved by the protocol. In order to get access to the access credentials (AC) exchanged between the two principals (P1) and (P2), the attacker must first, steal one of the devices or drive a cryptanalysis attack against the RC4 ciphered communication. For the first option the attacker will be discovered easily. In the case of the second one the attacker must possess sophisticated equipments, which makes it very unlikely to perform these kinds of attacks.

7. Implementation Consideration

In this section we propose where the protocol should be implemented in both platforms used for developing the SSO prototype

- The PICDEM FS USB Demo board device, which is a MPLAB IDE/C18 based development platform.
- The mobile phone, which is a NetBeans IDE 5.9 based development platform.

However, to understand these recommendations, the code produced during the development of the SSO prototype during the last thesis must be studied very carefully. The original code was not available in the previous thesis [52] but they are made available at the former supervisor Dr. Einar Sekness.

The integration of these new algorithm codes and packages to both development platforms should be done carefully in order to

- Ensure that the encryption and hashing used within the security protocol performs well when integrated into the prototype. Misplacing them or executing them during an incorrect time or instruction sequences will lead to improper implementation of the protocol. This can result in unknown security breaches.
- Avoid technical complications that may result from implementing these cryptographic algorithms into prototype.
- Keeps the prototype performing well i.e. keep it working with a reasonable speed.

In order to integrate properly the three cryptographic algorithms in the design of the SSO MIDlet, we have to always keep in mind two factors, which class and method is accessing the access credentials first when the user will accesses an account and last when he is quitting an account. The AES decryption method is always performed when a user accesses a new account. The encryption takes place when he quits that account. The access credentials must always remain encrypted in the (RMS) of the phone. In order to do so, the following has to be done:

- Install the Java Bouncy Castle Package, which offers full implementation of the AES-128, SHA-128. However, the RC4 algorithm is not free implemented in this package.
- Write a new dispatcher class, which calls the encryption and decryption methods of the standard AES algorithm.
- Execute the decryption and encryption before and after the access credentials are accessed and stored back on to the (RMS) of the mobile device. In SSO MIDlet the class that manges the (RMS) is called RMSmanagement. But the encryption and decryption should be called in two other possible classes, the Account logic or the Account. The first class is a manager class for all the accounts created by the user of the SSO MIDlet. The account class specifies the attributes and the variables of an account. How to call encryption to these two classes depends on the skills of the developer. Both classes have a reference to the RMSmanagement class, which opens, close, delete, save and add new accounts to the (RMS) of the mobile phone device.

In order to encrypt the access credentials with the RC4 before sending them through the Bluetooth radio-broadcasting channel to the Demo board device, the following needs to be done:

- Decrypt these access credentials and retrieve their plain text from the (RMS).
- Encrypt them with the RC4 stream cipher algorithm.

The RC4 ciphered access credentials should be then sent through the Bluetooth channel. The RC4 encryption method should be called in the AccountLogic or the Account class. These can perform the encryption by calling the same dispatcher class, which used fro the AES or we could use new dispatcher class for the RC4 and temporarily store the RC4 ciphered text in a vector and send it to the BluetoothTsaks class, which sends the ciphered text of the access credentials to the Demo board device through the Bluetooth channel. However, it is best to implement the encryption of the access credentials directely in the BluetoothTsaks class. However, due to the structure of the references of the classes in the SSO MIDlet, the implementation described above recommended.

The security protocol implements an authentication mechanism that authenticates a user to the SSO MIDlet. This mechanism is implemented in the Authenticate class. This class calls the SHA-1 Java Bouncy Castle algorithm through the dispatcher class. However, the SHA-1 hashing algorithm is used during the AES decryption/encryption and the RC4 encryption for key generation. The classes involved during these two process also call the SHA-1 hashing algorithm.

The RC4 should be integrated in the design of the Demo board device to make it able to decrypt the ciphered access credentials sent from the SSO MIDlet. The implementation of the RC4 should be in C18 as mentioned previously. The RC 4 should be integrated as an independent class to the design of this demo board device. Then the RC4 decryption method should be called to the class where the interrupt function is implemented. This is because this function reads the bytes received at the RX pin of the demo board and stores this data into a global circular buffer. The decryption should be executed when the number of bytes reaches 8 or more because after that, another function will be called to treat this data. However, if this data remains ciphered, the following functions will go blind causing the Demo board device to perform in an inappropriate way.

Another issue to be taken in consideration when integrating the protocol to the prototype is the mapping between the character and key codes. Performing this task, which is based on mappings between ASCII characters and their corresponding key codes using the HID usage table correctly [54], is highly recommended. This is because the username and password characters are sent to the Bluetooth/USB device from the SSO MIDlet as an 8 byte input report. However, this task is performed by the SSO MIDlet in the mobile phone device because of the superiority of its processing power and storage size compared to the Demo board device. We have to ensure that the RC4 encryption is invoked on the access credentials after this task is performed. If for some reasons the developers, which going to implementing the protocol have some doubts about this, then it is recommended that they implement it on the Demo board device. The disadvantage of implementing this task on this device is its limited storage and processing power, which could make the prototype work slower than the expected average. This would affect the usability results achieved early for the prototype in the previous thesis.

8. Discussion

The main goal for this master thesis was to secure the vulnerable SSO prototype developed in a previous thesis. There were three communication channels in the prototype that were not protected. This exposed the prototype to many security threats. During this work, a security protocol was developed to meet the security goals for the prototype listed as follow:

- First, protecting the Bluetooth radio-broadcasting channel between the demo board and the mobile phone device.
- Second, protecting the access credentials stored in the (RMS) of the mobile phone device.
- Third, providing a secure authentication mechanism that authenticates the SSO user to the SSO MIDlet.

However, designing a protocol that could meet all the above requirements for the SSO prototype was a very challenging task. This is due to the fact that the SSO prototype is composed of components with limited hardware power and that it has many different involved communications protocols, different developing platforms with many software packages restriction and incompatibility.

Protecting the first channel was considered the main task for the security protocol. In the protocol design, this channel is protected by encrypting it with the RC4 cryptographic algorithm. Some security experts may argue with this choice and may say that the RC4 is an outdated algorithm no longer recommended by the NIST in the designing of security protocols for wireless systems. There were some reported successful attacks against this algorithm. These were however, due to inappropriate use of the algorithm. The RC4 is only vulnerable when the method used for combining the secret encryption key and the initialization vector (IV) is not well designed and secured, or when its secret encryption key is 40-bits length or shorter as mentioned in section 2.1.2. Due to the limited processing power and memory size of the Demo board device simulating a keyboard in the prototype, we were looking for a secure and fast encryption algorithm. The RC4 was the best candidate that can fit for this task.

In the original design of the security protocol, the mobile phone and the Demo board devices should authenticate against each others. The authentication mechanism was based on sharing a secret password in advance, which is used for the authentication process between the mobile phone and the demo board devices.

Each of the devices uses this secret password for performing the authentication. After they successfully authenticate against each other, the devices construct the secret encryption key to be used for the RC4 encryption. This secret key must be identical, the original one is stored in the fixed memory of the demo board devices and the replica in the (RMS) of the mobile phone. This mechanism offers better authentication of the involved communicating principals against each others and a better method for securing the secret encryption key used for encrypting the Bluetooth communication. However, we had to tune this design and eliminate the devices authentication procedure and the generation of the secret encryption key by the Demo board device. These changes were needed due to the limited processing power of the demo board device that may not be able first, to execute a key exchange protocol and second, to compute correctly a secure secret key, which will be used during the RC4 decryption method. It was decided instead to save the value of this secret encryption key in a plain text to the demo board fix memory. This choice may cause some security concerns for the security experts. To address this issue, we tried

to hash the value of the secret encryption key before saving it into the fix memory of the Demo board. However, this was not successful due to the limitation discovered in the software developing platform MPLAB IDE/C18, which consisted in the lack of the necessary C functions library for implementing correctly the SHA-1 as mentioned in section 4.1.1.2 and also due too to the risk of complications that may arise from simultaneously implementing two cryptographic algorithms in a device with very limited processing power and memory size.

Another minor issue faced was related to the previous one is how to ensure a good generation of the secret encryption key. This key must be long enough to secure the data stream encrypted with the RC4 algorithm. The key generation function implemented within the security protocol generates the key out of the last 128-bits of the hashed value of the master password as mentioned in section 6.1 and stored in the (RMS) of the mobile phone device. This is a good generation method, which in average gives a good key length. However, we are not certain about the length of key or in other words, how many bits the Demo board could put to the RC4 decryption method. The developers recommend testing the maximum bits that could be allocated for this task by the Demo board device during the implementation phase of the security protocol. It is also recommend developing a checking mechanism that could limit the length of this secret key to the maximum bits that the Demo board device can offer to perform this task. This may seem as an ordinary task, but in reality it is important because based on the value of the secret encryption key, we can extend the length of the other protected secret key generated and installed in the (RMS) of the mobile phone device prior to the security protocol implementation. Knowing exactly the length of these keys will help the developers to implement correctly the security protocol into the design of the SSO prototype.

In the course of design of the authentication mechanism of the user to the SSO MIDlet and the protection mechanism for the access credentials stored in the (RMS) of the mobile phone device, the Java Bouncy Castle Package implementation for the cryptographic algorithms involved in the security protocol was evaluated with respect to security and reliability as mentioned in section 4.1.2 Only the SHA-1 and AES-128 are made free. The RC4 was not made available. The NIST claimed that the AES is secure even when implemented in its weakest version. Since it was correctly implemented no further attention was made to it. The implementation of the SHA-1 however, aroused security concerns. The fact that this cryptographic algorithm was implemented in its weakest version may be considered no longer secure because the security of SHA-1 has been somewhat compromised by cryptography researchers. The SHA-1 proved to have two major weaknesses:

- The file preprocessing step is not well designed to be a complicated task.
- There are certain math operations in the first round, which suffer from unexpected security problems.

These two major security weaknesses made the algorithm vulnerable to collision based attacks. The inventor of the attack believes that the stronger variant of the SHA, the SHA-2 is no longer considered to be secure either because it is algorithmically similar to the SHA-1. He also believes too that soon both variant the SHA-1 and SHA-2 will be subject to primage attacks [01].

The Java Bouncy Castle Package implementation only provides the implementation of the first variant of this algorithm, the SHA-1 and without its padding function. This implementation was not considered secure enough. However, for the protocol and due to the time limitation, an implementation to this algorithm is not provided. It was decided to rather focus more on the

design of the protocol than using time on some specific technical issue that doesn't represent the real goal for this thesis.

Another issue relevant for discussion is that after implementing the security protocol to the SSO prototype, the channel between the Demo board device and the PC will remained without any kind of protection. Leaving this channel unprotected yields many security concerns since it can expose the user of the SSO prototype to the many serious security attacks introduced early in this paper. During the design of the security protocol, securing this channel was not considered as a recommendation that must be met by the protocol. This decision was made in order to meet one of the main functionality requirements, i.e. the mobility requirement, in other words, to avoid restrictions on the log inn operation of the user on any PC or network.

This issue resulted from the contradiction of the main requirements of this SSO system with the fundamental security recommendations as mentioned in section 3.2. In order to overcome this problem, a hybrid solution was proposed for the SSO system mangers. Usually an SSO users use most of the time some specific PC, laptop or networks. If we protect the remaining channel between the Demo board and the PC by encrypting it with the RC4 too, the number of attacks dedicated to retrieving the access credentials of the SSO user during the log in process when the Demo board device sends them as an emulated key stroke to the PC will be substantially reduced. However, to keep this solution hybrid and meet the mobility requirement for the prototype at the same time, plug inn software has to be developed in such a way it makes the PC or laptop that the Demo board communicates with able to notify the Demo board on whether it is able to decrypt the RC4 stream of data . This software should be installed as a new plug-inn into the PC and should be integrated as a C18 code in the design of the Demo board device. This hybrid solution will provide more protection to the SSO user while keeping the prototype mobile. The SSO user should not be involved in taking decision on whether the communication between the Demo board device and the PC should be activated or not. He/she should however, decide during the first time whether to protect this channel with a specific PC or laptop. The rest of the process should remain transparent. Only the two communicating principals, the Demo board device and the PC, laptop or network should be involved in taking this decision.

9. Future Work

The prototype developed during a previous thesis was a successful one. It met all its functional requirements. However, this prototype was vulnerable to a number of security threats and needed to be protected before being able to be commercialized as a final SSO product.

Based on the previous achieved results, work and effort during this thesis was dedicated to secure this prototype. This was achieved by designing and analyzing a security protocol to ameliorate the security level of the prototype. During testing of the prototype prior to the design phase of the protocol, various technical complications were expected from implementing the protocol in the design of the prototype. This is in particular the case for systems such the SSO prototype, which has components with different hardware and software backgrounds, very limited hardware power, vulnerable communication protocols and limited software developing platform. Therefore, during the protocol design focus was always made on practical security solutions. However, this was a challenging task and not a job for novices.

Based on the results achieved during this thesis, the following listed future work is highly recommended. The suggested work has different security background. However, this is very critical in order to commercialize the SSO prototype.

9.1 Implementing the Security Protocol

In order to secure the prototype, the security protocol mentioned in chapter 6 has to be implemented into its design. This is not a complicated task though somewhat time consuming. The implementation of the three cryptographic algorithms involved in the protocol is recommended rather than usage of standard implementation available on the net. These algorithms should be adjusted and tuned to fit the technical specification of the mobile phone and the Demo board devices without affecting their security strengths. Implementations available in the free Java packages are standard implementation that do not fit easily and are processed on the MPLAB IDE /C18 platforms. Most of the standard implementations for these different algorithms for both platforms have proven to be weak, insecure and to a certain extent not professional. The implementation should be fully consistent with the protocol design. It is not recommended to allow for any changes or adjustment that may compromise the design specification of the protocol in order to overcome some technical design or platform related restrictions and limitations.

9.2 Implementing the Hybrid Solution

The Hybrid solution introduced in chapter 8 could enhance the overall security of the prototype. Therefore it is included in the further work. It is not a complicated task and doesn't require more than developing an extra plug-in software for the PC or laptop and injecting the same codes to the design of the Demo board device. The design of the Demo board should be configure to

- Decrypt the received access credentials to a plain text
- Convert it to emulated key strokes.
- Encrypt the emulated key strokes with the RC4.
- Send the cipher text to the PC.

The fourth step will not be performed in the case where the plug-in is not installed on the PC. The plug-in software installed on the PC should

- Decrypt the cipher text and retrieve the emulated key strokes.
- Forward the plain text of the emulated key strokes to the operative system of the PC.

After performing these two steps successfully, the operating system of the PC takes over and performs the log inn task for the SSO user.

9.3 More security Counter Measures

After implementing the security protocol in the design of the prototype as mentioned in chapter 7, there will still be unsecured channel in the prototype. This issue is out of the scope of this thesis. These channels need different security methods that can give improved and practical security counter measures than the counter measures provided by our security protocol. These particular channels that need improved protection are:

- The channel between the firmware provider for the Bluetooth/Demo board device and the purchased device used in the prototype.
- The channel between the SSO MIDlet server and the mobile phone device.
- The Channel between the firmware provider of the mobile phone and the purchased mobile phone used in the prototype.

It is best to secure these channels before commercializing the prototype since they can be prone to many attacks. Some of these attacks represent real security threats and could compromise the security of the prototype. If securing these channels is not possible, the SSO user should be made aware of the vulnerabilities resulting from leaving these channels unprotected by providing security guides or brochures.

9.4 Deploying the Original Hardware

The Demo board device used in this prototype as a keyboard emulator is not the original device. The original device developed by NISLAB is more compact and handy. Both devices have almost the same hardware technical specification and the same microchip. However, in order to integrate the original device into the prototype design, some changes must be conducted. The communication interface between the Bluetooth and the USB unit used in the existing prototype design needs to be rewritten. This is due to the fact that in the NISLAB device, the communication between the Bluetooth chip and the USB chip are performed internally in the same circuit board not externally as is the case in the Demo board device. The deployment of this component to the prototype before the implementation of the security protocol is recommended.

9.5 Put the sso Prototype Again to the Test

After the integration of the security protocol into the design of the prototype and the implementation of the other recommended future work described above, further tests on the prototype should naturally be conducted. The security counter measures developed during this work on the security protocol should be evaluated. Furthermore, the stability of the prototype after integrating the security protocol to its design should be assessed. These two goals could be achieved by carrying two type of tests:

- A practical penetration test where security experts should simulate the role of an attacker and try to:
    - Get access to the access credentials stored in the (RMS) of the mobile phone device without knowing the value of the master password.
    - Retrieve the access credentials sent from the mobile phone to the demo board devices through the Bluetooth broadcasting radio channel.
- A usability test to evaluate how stable the prototype became after implementing all the new security measures to it. This test will reveal whether the prototype kept its functionality requirement, which were met before implementing the security protocol, or that it has scarified some of them for achieving more security.

These two kinds of tests are highly required before one can claim that the prototype is a final product and start commercializing it.

10. Conclusion

In this chapter, the questions proposed in section.1.4 will be answered. These questions were based on the main security requirements that should be taken in consideration when designing a security protocol for protecting the SSO prototype.

How should the design protect the access credentials from the man in the middle attack?

The access credentials are exposed in the prototype to such attacks when they are sent from the mobile phone to the Bluetooth/Demo board device through the Bluetooth radio-broadcasting channel. This is due to the fact that the Bluetooth protocol itself suffers from many security weaknesses as mentioned earlier in this thesis as mentioned in section 2.3. This made the Bluetooth vulnerable to many kinds of attacks including the man in the middle attack, which represents a great challenge for the prototype. However, in order to protect this channel from such attacks a solution based on encrypting the Bluetooth channel with the RC4 algorithm was developed as mentioned in section 5.3.1. The solution represents the best practical solution that can best fit the hardware power of the devices involved in this communication.

How to protect the access credentials stored in the (RMS) *of mobile phone device?*

In the previous prototype the access credentials were stored in the (RMS) of the mobile phone device in a clear text without any sort of protection. This represents a major security issue making the prototype vulnerable and exposing its users to a number of serious security threats. In order to overcome this problem, we were looking for practical solutions. Although the mobile phone device has relatively better hardware power in comparison to the Demo board device, it was still not able to perform complicated and strong cryptographic computations. For instance all the cryptographic algorithms implemented by the Java Bouncy Castle Package or other packages were implemented in their weakest form as mentioned in section 4.1.2. Furthermore, some of them were subject to some changes that weakened them even further to the extent that they could no longer be considered secure. An attempt was made prior and during the design phase of the protocol to seek ready implementations for the involved cryptographic algorithms to help design a practical solution.. However, no free of charge implementation better than the one already offered by the Java Bouncy Castle Package was found. For the future, it is recommended to use more reliable implementations. The protection mechanism for the access credentials was based on encrypting the access credentials of the accounts with the AES-128 before they are stored in the (RMS) of the mobile phone as mentioned in section 5.2.2. Every time a user accesses one of his/her accounts, the security mechanism generate first a short-term secret encryption key based on hashing the account name with the master password as explained early. Every account should have its own unique short-term secret encryption key, which must remain discreet for the user. When the user is accesses the account, the system generates the secret key, then decrypt the access credentials and retrieve the plain text. After using the account, the access credentials will be encrypted and the cipher text will be stored in the (RMS) of the mobile phone. It should be noticed that the security mechanism is activated automatically and remains transparent for the user of the prototype.

How should the SSO user securely authenticate him/herself to the SSO MIDlet, which is installed in the mobile phone device?

The SSO MIDlet in the prototype provided a simple authentication mechanism, which represented the only line of defense for avoiding an attacker from getting access to the access credentials that are stored in the (RMS) of the mobile phone. However, this was not satisfactory. It was decided to provide more sophisticated authentication mechanisms, which will represent the first line of defense but not the last one, which an attacker must compromise before getting access to these access credentials. The authentication mechanism as mentioned in section 5.2.1 is based on using only the first 32-bits of the hashed value of the master password. We are using only 32-bits for the authentication purpose because the generation of the long-term secret keys recommend the use of the remaining 128-bits. This hashed value is also used in the construction of other short-term secret keys used at later stage of the protocol as described earlier. The master password is the only key recommended for the authentication of the user to the SSO MIDlet. This authentication mechanism enhances the overall security of the prototype. However, it should be mentioned that the SHA-1 is not longer considered a secure hashing algorithm and that the implementation from the Java Bouncy Castle Package is performed without the padding function. This is not encouraging factors however, this doesn't represent a serious problem since during the implementation phase one could change this implementation with a better one. There are no restrictions on changing the implantation or the algorithm itself by another one so long it operates in the 8-bits mode. The security protocol developed here represents a framework for the cryptographic algorithms. These algorithms are no more than components that could be changed with ease. The only consideration that the developers should keep in mind is to continue work only with 8-bits implementations due to the limitation of the hardware power of the devices used by the prototype, which does not allow for more computation than this.

How should the security protocol authenticate the mobile device and the Demo board with each other?

When work started on the thesis, it was thought that every one of three algorithms used in the security protocol could be implemented in the design of the Demo board device. However, careful study of the technical specification of the developing environment, the MPLAB IDE used for developing software for this kind of microchip, it was revealed that that the function library of the C18, which is the C base for the IDE was very limited and did not support any of the cryptographic library as mentioned in section 4.1.1.2. Attempts were made to overcome this particular problem by using IDE base C30. However, the C30 was for 16 and 32-bits implementations and its cryptographic library is not made available free of charge. Due to these limitations, it was decided to abandon the original design, which was based on implementing a simple hand check protocol between the mobile phone and the Demo board device where both devices confirm the identity of each other prior to the start of the communication and after establishing the Bluetooth communication between the mobile phone and the Bluetooth/Demo board devices. Instead, another approach was adapted for solving this issue. The new approach gave good security results. It consisted in applying a solution based on installing an identical secret key in both the mobile phone and the Demo board devices as mentioned in section 5.3.1. The identical secret key is used for the RC4 encryption/decryption of the access credentials stream data and at the same time as an authentication mechanism for both devices. The (Ek2), which is generated from the last 128-bits of H(mpw) XORed (k2) is stored in the mobile phone. The secret key (k2), which is a one-time key is generated prior to the (Ek2) and is installed in the Demo board device. When the two devices start communicating, the Demo board decrypts the communication with this key (k2). However, if it is a face device, then the user will not be able to login to his/her account. After one or two login attempts he/she will realize that they are communicating with the wrong Demo board device. For verification, the inverse equation can be used to find out whether the (k2) key was correct. The inverse equation is (k2) = last 128-bits of H(mpw) XORed (Ek2). The use of the two identical secret keys in both devices proved to be an

adequate solution to overcome the technical limitation and provided level of security similar to the one expected by implementing hand check protocol into the security protocol design.

How and where should the security protocol in the design of the SSO prototype be implemented without affecting the usability or stability of the prototype?

The implementations of the algorithms should be integrated in two different design parts as mentioned in chapter 7. Therefore the algorithms used by the protocol should be written in both languages: Java and C18. The implementation specification of the same algorithm should be identical for both platforms otherwise one can risk facing technical incompatibility problems. E.g. an implementation for the RC4 with different generator method for the IV or which uses different key length for the encryption/decryption methods could not be provided.

One has to ensure that the RC4 encryption is invoked on the access credentials data by the mobile phone device after performing the mapping between the character and key codes and before the stream of data is sent through the Bluetooth radio-broadcasting channel. This task could be implemented in the Demo board device instead. However, it is more appropriate to implement it in the mobile phone to benefit from extra hardware power. If this task is not implemented correctly the system goes blind. This is due to the fact that if the access credentials are encrypted before the character mapping is performed on the mobile phone device and the Demo board doesn't support this function then the PC will not understand the data and the log in operation will not be performed.

These were the major considerations the developers should keep in mind when integrating the protocol to the prototype.

What are the alternative solutions in case the design is not be able to provide an adequate security level due to limited processing power of the devices used in the SSO prototype and due to limited software development environment of both platforms that may will force us to use weak cryptographic algorithms?

During work on this thesis many technical problems were encountered. These were due to limitations of the two main devices used by the prototype: the Demo board on a major level and the Mobile phone on a minor level. The technical challenges related to the limited hardware power of the mobile phone device and the Java Netbeans IDE environment were manageable and did not require radical changes to the design of the security protocol as mentioned in section 4.1. However, hardware limitations of the Demo board device along with the limitation of the MPLAB C18 library functions required several radical changes to the design of the protocol.

- The initial plan was to encrypt the Bluetooth radio broadcasting channel with the AES-128. But this was abandoned and it was decided to use the RC4 instead.
- A hand check protocol was to be implemented prior to the authentication so that the two communicating devices could be authenticated. However, this was also abandoned for the reasons listed earlier.
- The limited processing power of the Demo board device restricted the length of the long-term secret key and the key generation mechanism of the protocol.

Fortunately, the security analysis performed on the security protocol proved that it was practical and secure for to be implemented into the prototype. We were seeking too for a good implementation of the three cryptographic algorithms used within the protocol. This could be achieved by putting extra time and effort to the project for implementing these algorithms and by

putting extra funds for getting the appropriate developing software and security packages for both platforms.

The security protocol developed during this thesis could be regularly updated in order to maintain its performance in protecting the prototype or when new versions of the prototype are released. It represents a security frame work, with different security mechanisms organized together to provide a good security level for the user of the SSO prototype. Updating the protocol with new implementations of the cryptographic algorithms will not recommend any radical changes on the original design of the prototype. If in the future the developers overcome some of the limitation encountered in this work and are able to implement better algorithms on the MPLAB IDE platform, then it is recommended to update the security protocol with an authentication mechanism that could authenticate the mobile phone and the Demo board devices against each other.

Finally, the security protocol developed during this thesis will add an extra value to the prototype. The protocol offers secure and practical solutions for most of the technical security issues that the protocol was prone to. Its design now complies totally with the main recommendations specified for the SSO prototype. The hardware, protocol, platform technical limitations related to the different component used by this prototype are resolved without any radical changes to the design of the prototype. The protocol is designed to be easily integrated in the design of the prototype while keeping it stable and functional.

Bibliography

[01] Alfred Menezes, Paul C. van Oorschot, and Scott A. Vanstone, Handbook of applied cryptography, CRC Press, 1996

[02] Security In Fixed and Wirless Networks, Gunter Schafer, wiley, 2003

[03] Wikipedia. January 2006. Advanced encryption standard. http://en.wikipedia.org/wiki/AES (Last visited May 2007).

[04] Wikipedia. January 2006. RC4. http://en.wikipedia.org/wiki/Rc4 (Last visited May 2007).

[05] Wikipedia - Avalanche effect. http://en.wikipedia.org/wiki/Avalanche_effect. (Last visited May 2007).

[06] Wikipedia - "Cryptographic hash functions". http://en.wikipedia.org/wiki/Cryptographic_hash_function. (Last visited May 2007).

[07] Pashalidis Andreas and Mitchell Chris J. A taxonomy of single sign-on systems, information security group, Royal Holloway, University of London. *In Lecture Notes in Computer Science*, volume 2727:pages 249–264, July 2003

[08] Lastbit Software. Password director. http://www.passworddirector.com. (Last visited May 2007).

[09] CEZEO Software. SecureWord Mobile. http://www.cezeo.com/products/secureword-mobile. (Last visited May 2007).

[10] Inc Com Consulting. Mobipassword. http://www.mobipassword.com. (Last visited May 2007).

[11] ESoftPRO. Online password manager. http://www.handyarchive.com/Utilities/Password-Recovery/12343-Online-Password-Manager.html. (Last visited May 2007).

[12] MyPasswordManager. Your powerful password keeper. http:// mypasswordmanager.com/index.htm. (Last visited May 2007).

[13] Pashalidis Andreas and Mitchell Chris J. A single sign-on system for use from untrusted devices, information security group, Royal Holloway, University of London. 2004.

[14] Samar Vipin. Single signon using cookies for web applications. in. *Proceedings of IEEE 8th International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*, pages 158–163, June 1999.

[15] Pashalidis Andreas, Mitchell Chris J. Single sign-on using trusted platforms. technical report RHUL-MA-2003-3, Mathematics Department, Royal Holloway, University of London. 2003.

[16] Microsoft Coperation. Microsoft passport. http://www.passport.com. (Last visited May 2007).

[17] Pashalidis Andreas, Mitchell Chris J. Using GSM/UMTS for single sign-on, information security group, Royal Holloway, University of London. 2003.

[18] Protocom. Securelogin. http://www.protocom.com/html/securelogin_single_sign_on.html. (Last visited May 2007).

[19] Niemi Antti T. Single sign-on with Kerberos V, Helsinki University of technology. http://www.tkk.fi/cc/docs/kerberos/sso.html, 2002. (Last visited May 2007).

[20] Satoh Fumiko, Itoh Satoh. Single sign-on architecture with dynamic tokens, IBM Research, Tokyo Research laboratory, Japan. *Saint Symposium on Applications and the Internet*, page 197, 2004.

[21] Wireless network security 802.11, Bluetooth and handheld devices http://csrc.nist.gov/publications/nistpubs/800-48/NIST_SP_800-48.pdf (Last visited May 2007).

[22] Herfurt Martin. Bluesnarfing @ CeBIT 2004, detecting and attacking bluetoothenabled cellphones at the Hannover Fairground. *Salzburg Research Forschungsgesellschaftmb*H, *Austria*, 2004. http://trifinite.org/Downloads/BlueSnarf_CeBIT2004.pdf (Last visited May 2007).

[23] Shaked Yaniv and Wool Avishai. Cracking the Bluetooth PIN In. *Proceedings of the Third Annual International Conference on Mobile Systems, Applications and Services: MobiSys*, pages 39-50, June 2005.

[24] Levi Albert, et.al. Relay attacks on Bluetooth authentication and solutions. *Lecture Notes in Computer Science*, 3280:pages 278– 288, January 2004.

[25] Microchip. PICDEM FS USB demonstration board users guide. 2004. http://ww1.microchip.com/downloads/en/devicedoc/51526a.pdf (Last visited May 2007).

[26] Free2Move. Bluetooth serial port plug - F2M01C1 Datasheet. 2004. http://free2move.se/pdf/Datasheet_F2M01C1.pdf (Last visited May 2007).

[27] Microchip. MPLAB C18 C compiler libraries. 2005. http://ww1.microchip.com/downloads/en/DeviceDoc/MPLAB_C18_Libraries_51297f.pdf (Last visited May 2007).

[28] Utimaco Software. SafeGuard Advanced Security. http://www.utimaco.com/C12570CF0030C00A/CurrentBaseLink/W26K9K2R406OBELUK. (Last visited May 2007).

[29] SunMicrosystems. The Java 2 Standard Edition (J2SE) v1.4.2 2006 http://java.sun.com/j2se/1.4.2/download.html (Last visited May 2007)

[30] SunMicrosystems. The Connected Limited Device Configuration (CLDC) v1.1. 2006. http://jcp.org/aboutJava/communityprocess/final/jsr139/ (Last visited May 2007).

[31] SunMicrosystems. The Mobile Information Device Profile (MIDP) v2.0. 2006. http://java.sun.com/javame/index.jsp (Last visited May 2007).

[32] SunMicrosystems. The J2ME Wireless Toolkit. 2006. http://java.sun.com/products/sjwtoolkit/ (Last visited May 2007).

[33] SunMicrosystems .The NetBeans Mobility Pack. 2006 http://www.netbeans.org/kb/articles/mobility.html (Last visited May 2007).

[34] Ortiz C. Enrique. Using the Java APIs for Bluetooth Wireless Technology, Part 1- API Overview. (Last visited May 2007). http://developers.sun.com/techtopics/mobility/apis/ articles/bluetoothintro/, 2004.

[35] Mahmoud Qusay H. The java APIs for Bluetooth wireless technology, part II. *Sun Developer Network*, 2003. http://developers.sun.com/techtopics/mobility/midp/articles/bluetooth2/ (Last visited May 2007).

[36] USB implementers' forum. USB Device class definition for human interface devices (HID) Firmware specification version 1.11. 2001. http://www.usb.org/developers/devclass_docs/HID1_11.pdf (Last visited May 2007).

[37] Jakobson Markus and Wetzel Susanne. Security weakness in Bluetooth In D. Naccache, editor,Proceedings of the Cryptographers' Track at RSA Conference,. *Lecture Notes in Computer Science*, volume 2020:pages 176–191, April 2001.

[38] Levy Ophir and Wool Avishai. A uniform framework for cryptanalysis of the Bluetooth E0 cipher. *Cryptology ePrint Archive*, Report 2005/107, 11 April 2005.

[39] Shaked Yaniv and Wool Avishai. Cracking the Bluetooth PIN In. *Proceedings of the Third Annual International Conference on Mobile Systems, Applications and Services: MobiSys*, pages 39-50, June 2005.

[40] Microchip. The dsPIC30F cryptographic library. 2006 http://ww1.microchip.com/downloads/en/DeviceDoc/51468B.pdf (Last visited May 2007).

[41] Microchip. The MPLAB C30. 2006 http://lsa.epfl.ch/education/courses/MicroInfo/doc/51284D.pdf (Last visited May 2007).

[42] SunMicrosystems. 2005. J2se: Security and the java platform. http://java.sun.com/security/ (Last visited May 2007).

[43] Knudsen, J. 2001. *Wireless Java: Developing with Java 2, Micro Edition*. Apress.

[45] BouncyCastle. Bouncy castle crypto package. Light-weight API, release 1.31. http://www.bouncycastle.org (Last visited May 2007).

[46] IAIK. Jce-me. Light weight cryptography library. http://jce.iaik.tugraz.at/sic/products/mobile_security. (Last visited May 2007).

[47] Corporation, P. T. 2005. Phaos micro foundation. Light weight cryptography library. http://www.phaos.com/products/crypto_micro/pmf.html. (Last visited May 2007).

[48] NTRUCryptosystemsInc. Ntru neo for java. http://www.ntru.com/products/Neo_Java1.pdf. Cryptography on Java devices. (Last visited May 2007).

[49] Tillich, S. & Großschadl, J. A survey of public-key cryptography on j2me-enabled mobile devices. Graz University of Technology, Institute for Applied Information Processing and Communications Inffeldgasse.

[50] Olsen Ole Kasper. Adversary modelling. Master's thesis, Gjøvik University College,2005.

[51] Egeberg Tommy. Storage of sensitive data in Java enabled cell phone. Master's thesis, Gjøvik University College (GUC), 2006.

[52] Byfuglien Mats. A mobile single sign-on system. Master's thesis, Gjøvik University College (GUC), 2006.

[53] A. W. Dent and C. J. Mitchell. *"User's Guide to Cryptography and Standards*", Artech House Computer Security Series, 2004, chapter 11-12, pp. 215-266.

[54] USB implementers' forum. USB HID Usage Tables v1.12. 2005. http://www.usb.org/developers/devclass_docs/Hut1_12.pdf (Last visited May 2007).