Implementation of SMS Application with Mutual Authentication and Perfect Forward Secrecy

James Brotherston *(jamesbrotherston@gmail.com)*, Colin Hilchey *(chilchey@interchange.ubc.ca)*,
Kelvin Tsui *(legokel@gmail.com)*, George Wang *(grizby@gmail.com)*

***Abstract*— Design specifications, prototype application description, usage instructions, and guiding principles of a cell phone application used to implement an additional layer of security for SMS messages. Current vulnerabilities of messages sent over the GSM network stem from SIM card readers capable of retrieving stored and deleted SMS messages, and also from invisible software that can be installed on some cellular phones. To address these vulnerabilities, an application handles message security, and avoids plaintext being stored on the SIM card or in the phone memory for long periods. Instead, plaintext is held in temporary application-level memory. The key used for message transmission is established using ephemeral Diffie-Hellman key exchange, where a shared key between two parties is used for authentication and encrypting the exchange of Diffie-Hellman keys. The time and power required to generate the public keys may be a limiting factor in usability. Appropriate usage scenarios are described for the application, and includes details of how users can use the program. Some problems are associated with the application's design and implementation, and future work will address many of these issues.**

***Index Terms*—Cell phone security, SMS messages, Cryptography, Perfect Forward Secrecy**

I. Introduction

This document outlines the design of an application for cell phones that can be used to improve confidentiality and integrity of SMS messages sent over the GSM network. To improve and build upon existing capabilities that can be implemented on phones, the designed application focuses on establishing a relatively secure path for short text communications of sensitive material between two parties with a shared symmetric key.

Current threats to users of SMS messages are outlined first, specifically where threat agents are people who may have access to a user's phone and/or SIM card for a short period of time. With those vulnerabilities in mind, the next sections outline some existing countermeasures which improve the secrecy of messages sent over the cellular network.

Assumptions for the appropriate use of the proposed cell phone software are stated explicitly before describing in detail the connection, key establishment, and message passing algorithms.

The authors developed a prototype application, the details of which are presented following the design details. Some aspects of the implemented program are relatively complete and other aspects need improvement and optimization before the system is ready for large distribution. Any problems that the authors discovered during the development process are discussed, not limited to timing details and potential usability issues.

Future work and proposed solutions for the problems are then presented, especially in the context of how to build on the existing application.

Finally, to summarize the benefits and improvements offered by the proposed design, principles of designing secure systems are briefly discussed as they pertain to the algorithms and protocols to be implemented in this project.

II. Existing Threats to Message Security

In this section, threats to users of cellular messaging are discussed. The paramount risk is that data meant to remain confidential between two parties may be read by people/systems other than the intended recipient. This breach of confidentiality may give rise to a threat of misuse of information held in the messages.

The information held in text messages can be any sort of raw data limited to 160 characters, but to illustrate the sensitivity of some data, some examples are appropriate. In particular, messages may contain personal information, company trade secrets, contact information for friends or relatives, and passwords used for any number of context. One-time passwords are sometimes transmitted to users of other systems using the SMS protocol through systems such as that in [1]. An interception of such a message could lead directly to an intrusion of a password recipient's account on whatever system the password is for. Other personal or company details could be of significant use to identity thieves, rival companies, burglars.

Manuscript received December 7, 2009. This paper is the final report for the authors' term project in EECE 412 101, overseen and taught by Professor Konstantin Beznosov.

James Brotherston, S81373060, is in his third year of a B.A.Sc. Program of Computer Engineering at UBC (jamesbrotherston@gmail.com).

Colin Hilchey, S55009054 is in his fifth year studying toward a B.A.Sc. in Engineering Physics at the University of British Columbia, specializing in the Electrical option. (phone: 604-288-5615; e-mail: chilchey@interchange.ubc.ca).

Kelvin Tsui, S64559065, is in his 3rd year of B.A.Sc. Computer Engineering at UBC. (e-mail: legokel@gmail.com)

George Wang, S77243996 is in his 4th year of B.S. In Computer Engineering at UBC. (e-mail: grizby@gmail.com)

In each of the specific threats mentioned below, once the SIM card or phone has been attacked, the attacker will have essentially full access to message plaintext and other details about the messages.

A.Software for Message Forwarding/Uploading

Immediate threats to confidentiality exist when software such as that available in [3] for Android phones, and [4] for a large assortment of Nokia phones. The software claims to be completely invisible and untraceable, so the user of the phone on which the software is installed would have no idea their messages are being read by another party. There is an assortment of products available for different ranges of phones and with slightly varying implementations, but most claims to track details about both ends of the communication, as well as any message. Some packages, such as that in [3], will upload any message sent or received from the infected phone to a centralized server where the attacker can read the messages indefinitely into the future. Other packages, such as those in [4], can forward any incoming and outgoing messages to the attacker's phone.

For software such as that discussed above, messages could continue to be read by third parties indefinitely into the future.

B.SIM Card Readers

SIM Card readers are available for purchase that can read any messages that have been sent and stored on the phone users' SIM card. An example product can be seen in [2], which even claims to be able to recover messages which have been deleted from the SIM card. The resulting threat is that anyone with access to a user's phone for a few minutes can copy any SIM card data onto a computer and mine the information later on. Once these messages are known, many contact details and any text messages that were ever stored on the SIM phone are potentially at risk of becoming public information, and any information in those messages could be misused in ways not limited to those mentioned above. When the SIM card has been sacrificed, only messages from the past can be seen, but depending on the stealth and regularity of the attacker, the same SIM card could be obtained for short periods to remain relatively up to date with the communications of the cell phone's owner.

Other higher cost alternatives to eavesdropping by means of cracking GSM encryption are likely to become commercially available as well, as discussed in [5]. These methods would require no access to the phone, and less than a half hour of time in order to obtain the plaintext of SMS messages.

III.EXISTING COUNTERMEASURES

Some existing countermeasures are available for cell phone users to reduce the risk of lost data confidentiality and other threats stemming from such breaches.

A.Software for Mobile Phones

Some software packages can be installed for allowing users to encrypt and decrypt messages using a range of cryptosystems for assuring confidentiality and message integrity. Such software can be found in [6] and [7].
Software discussed in [6] uses elliptic curve cryptography for improved efficiency, and charges money for each enciphered message sent. Diffie-Hellman key exchange is used in this system, but the keys associated with different users are stored in a keyring that appears to be used indefinitely.

Software discussed in [7] is open-source and freely available, but does not implement Diffie-Hellman key exchange. Rather, it uses the PGP infrastructure for public and private key pairs.

B.Phones made for additional security

Some commercial phones are available which place emphasis on the crypto associated with them. These phones include Diffie-Hellman key exchange capabilities as well as a variety of other crypto, but their appeal is to a different group with higher security needs than the target of this project. Details of an example phone built with crypto in mind can be found in [8].

IV.ASSUMPTIONS FOR EFFECTIVENESS OF ADDED SECURITY

Given the particular threats addressed in the 'existing threats' section, and the existing countermeasures available to the public, this project focuses on solutions for users in scenarios given the following assumptions for the proposed design:

- A shared value/passphrase can be established between the two parties wishing to communicate through text messages
- Extra cost is willing to be exchanged for enhanced messaging security, as each user needs to send up to seven messages during authentication and key establishment
- Both users have a phone with sufficient memory and processing to perform Diffie-Hellman key exchange operations within a reasonable amount of time.
    - The testing section contains details of how much time is required for parameter generation
- Both users are able to install a version of the application on their phone
- Each user issues both a challenge and response for authentication purposes, and if the authentication is not completed, no messages can be passed using the message application

V.APPLICATION DESIGN

This section outlines the application designed and implemented by the authors. It was designed for the Android phone platform 2.0, and tested on an emulator
Note that for the following discussion, 'A' is the person who would like to send a secure message to 'B.'

A.Usage Scenario

This system for securing text is meant to work when both parties are aware of the assumptions listed in the above section, and where each knows that from time to time the other may send a request to send a secure message.
.

B.Sequence of activities

The authentication and key establishment procedures are summarized in Figure 1 below.

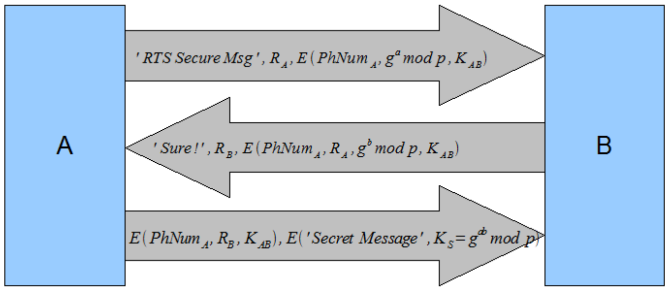
A
B
$'RTS\ Secure\ Msg', R_A, E(PhNum_A, g^a \bmod p, K_{AB})$
$'Sure!', R_B, E(PhNum_A, R_A, g^b \bmod p, K_{AB})$
$E(PhNum_A, R_B, K_{AB}), E('Secret\ Message', K_S = g^{ab} \bmod p)$

Figure 1. Authentication and key establishment in DFCS Messaging application

The full protocol, including what users A and B must do, is as follows:

- A and B agree on a shared secret value, from which their private shared key, KAB, will be derived using a cryptographic hash function

- A opens the phone application for secure text messages, and selects a contact from the main screen of the application, seen in the figure below. If the contact is not already stored on the list, the user can add a new contact at any time.

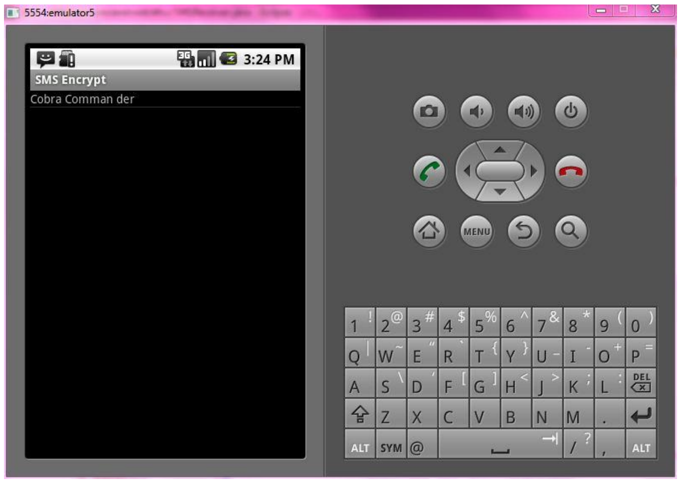

Figure 2: Main Screen, Contacts List in DFCS Messaging application

- A enters the appropriate passphrase/shared secret. Setting the password sends a challenge to B. See Figure 3 for what this looks like to the user.

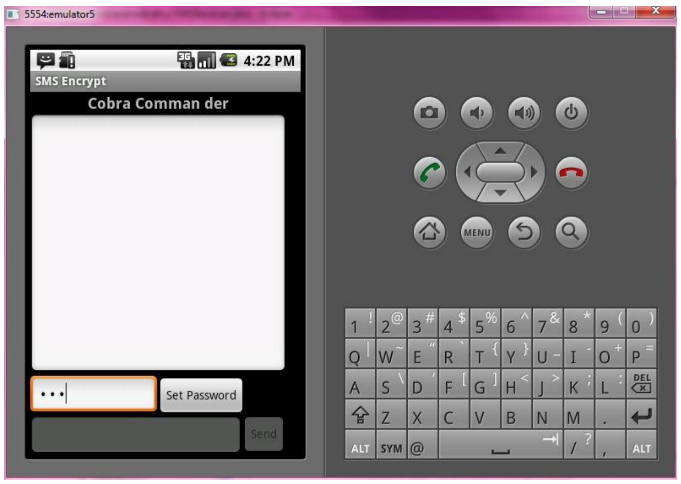

Figure 3: Setting the password

Setting the password in the application does different activities for A and B. For A, setting the password does the following:

- Generates a public and private Diffie-Hellman key pair. The parameters g and p are each between 512 and 1024 bits.
- Generates a nonce
- Sends authentication request to the selected recipient

Once B receives the challenge, he/she will select the appropriate user from the contact screen. B then enters the password, which initiates the following activities in the application:

- Decrypts the public Diffie-Hellman key
- Generates a public and private key based on the parameters received from the other party
- Calculates the session key, $(g^a \bmod p)^b \bmod p$
- Encrypts the nonce and the public DH key
- Generates a new nonce as a challenge
- Sends the full response, i.e. the encrypted nonce R_A, as well as the challenge nonce

A now receives B’s messages, and the application does the following:

- Parses B’s message, decrypts B’s response, and parses the public key from the decrypted response
- Authenticates B by checking the nonce and phone number, as seen in Figure 1
- If authentication is successful, A will be told that B is authenticated, as seen in Figure 4
- Calculates the session key, $(g^b \bmod p)^a \bmod p$
- Encrypts B’s nonce and A’s phone number with the shared key

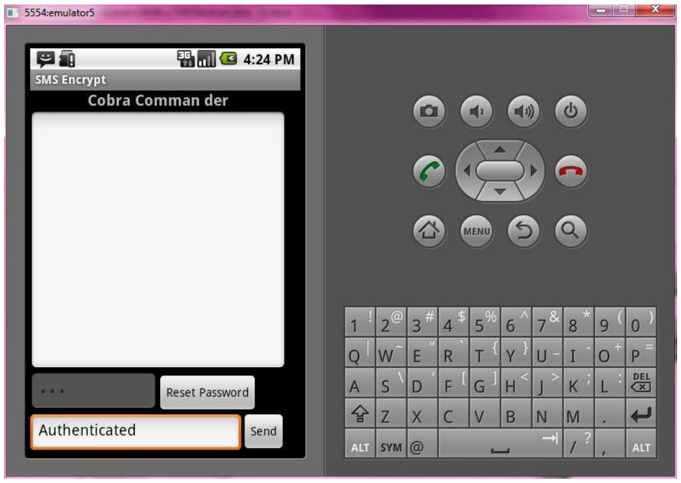

Figure 4: Letting the user know authentication is successful

A can now enter a message to send to B. When he/she hits the send button, the response message and the session message will be sent to B.

B can then authenticate A by checking that the nonce, and, assuming authentication is successful, he can trust the message encrypted with the session key. If authentication is unsuccessful, the 'decrypted' message will look like gibberish anyway.

A secure session is now open between the two users, and they can send messages back and forth in the same way as an instant messaging application.

When the session is done, the session keys are deleted, as are the exponents used for the key exchange. In the future work section of this paper the authors discuss timeout functionality, which will protect the message confidentiality even if a user mistakenly leaves a session open.
Once the session key is deleted, there is no way of reading the messages that were sent between the parties even if the shared key is broken.

C.Novel aspects of the design

Although additional security of text messages with phone applications is not an entirely new concept, the authors did not find any other messaging systems with perfect forward secrecy. The messages can be effectively destroyed once the session is complete, and a breach of the shared key between two users does not sacrifice confidentiality of past messages.

VI.MESSAGE PARSING

To facilitate proper message interpretation and parsing, the application must add appropriate headers onto messages so that the recipient's application is able to reassemble multiple messages and parse parts of the message according to what needs to be done. Headers are used to facilitate the following:

- Differentiate between regular messages and messages intended for DFCS Messaging application
- Differentiate between messages as part of a handshake, and messages meant for human eyes
- Differentiate between new messages and messages which are continuations of previous messages
- Introduce the nonce and key exchange details, so that the program can determine at run-time where the different data is contained the message

When a message is received by the application, the headers are removed from the messages, and if it is a multi-part message then the message will be reassembled before being decrypted.

VII.APPLICATION FLOW

Whereas the Application Design section above details what the application does, this section shows what the application's classes are doing throughout the authentication and session periods.

Figure 5 shows the sequence of activities on the application of the initial message sender. Figure 6 shows how the initial receiver's application handles authentication, and Figure 7 shows how all subsequent session messages are managed.

Note that SMS, SMSConvo, SMSCrypto, and SMSTransmitter are classes in the application which handle the actions as seen in the sequence diagrams.

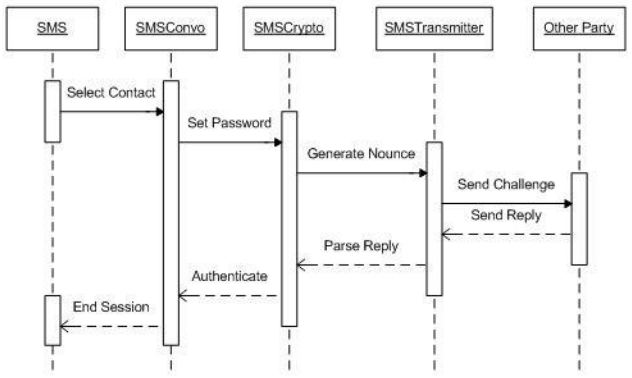

Figure 5. Sequence diagram for application, from point of view of sender

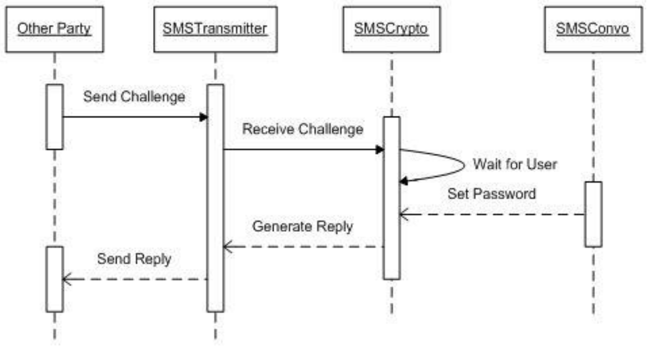

Figure 6. Sequence diagram for application, from point of view of challenge receiver

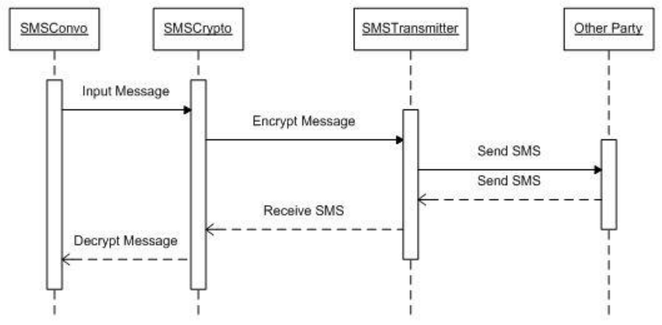

Figure 7. Sending messages once session key has been established

VIII. Testing, Results, and Problems Encountered

There are a number of issues that arose during development and testing of the application.

Computation of Diffie-Hellman parameters may take anywhere from 2 to 10 minutes (takes average of 8 minutes on the emulator). This parameter calculation is computationally expensive and might draw a lot of power, thus using up a lot of battery power. The Android platform emulators do not emulate battery consumption, so specific data on consumption is not available.

The size of the Diffie-Hellman parameters can range from 512 to 1024 bit numbers. The passing of these parameters that initiates the handshake will take several messages to transmit as each SMS message can only take 160 characters. The user that sends the challenge may have to send a total of seven messages, and the receiver has to send one message in reply to complete the cryptographic handshake. If the users do not have an unlimited SMS messaging service it can become quite costly for the user.

The number of messages sent may be reduced if the authors are able to send messages through the SMS Data channel instead of the standard SMS text that only accepts characters. It might also be possible to apply simple compression to the text messages.

The user may have many contacts that they want to SMS securely with. With the current implementation the user is required to remember passphrases for each of their contacts. This would make remembering passwords increasingly difficult and decrease usability. This also may result in the user and contacts using duplicate passwords. This could make communication possibly very insecure. A solution to this problem could be the use of a password protected keyring that contains the keys to each contact. Although this reduces the point of attack to a single master password, it makes managing large contact lists possible.

IX. Breaking Message Security

There are certain issues and possible vulnerabilities that need to be addressed regarding the current application implementation.

One major point of attack is on the user password(s). If an attacker is able to get the shared passwords, authentication is compromised as the attacker can pose as the user. The conversation logs that the user may keep would be compromised as well if it is encrypted with the same password.

A keylogger installed on the phone is one way to get the user passwords, but it can also get the messages sent by the user from the logged keystrokes. However, incoming messages are still secure as keyloggers only grabs data from the user’s keystrokes.

One possible way of getting access to the conversation and application data such as session keys, etc are with memory dumps. While the program is running, the conversation and session key is stored in memory, and only cleared upon exiting the application. Memory dumps could reveal cryptographic keys and unencrypted messages. This, however, is part of memory management on the Android OS and how it handles the memory for each individual application. Further testing and analysis would be required to determine the possibility of this attack.

While the message contents are encrypted, the SMS message header is not. Due to this restriction, the user cannot encrypt who they are sending the messages to or where the messages originate from.

The current implementation of message encryption is done with AES in counter mode with no padding. The lack of padding allows an attacker to approximate the length of message sent. This may compromise short or one word messages.

X. Future Work

The future developments of the project may include the following:

- Session timeouts, automatic deletion of session key
- Backlogging messages as user option
- Use of a keyring
- Sending bytes through a data message instead of hex characters as text
- Improving efficiency of software implementation

Since a user could leave their phone unattended with a session still running, a session timeout would greatly improve security. The session keys should be deleted after a certain period of inactivity.

When encrypted messages are received and decrypted for the user, he/she should be able to store the messages encrypted with a different key than is shared with another party. This feature would introduce new security risks, but is a feature that users may desire. If users are not satisfied with the features available in the application, they will simply opt not to use it and not use additional protection for their messages.

In the current implementation, encrypted messages are converted to hexadecimal characters before being sent. This ensures that there will be no errors with displaying or sending non-text characters, and was chosen primarily to ease the proof-of-concept development. Since this doubles the length of sent ciphertext, a future improvement will be to send the bytes of ciphertext as a data message rather than as a text message.

XI.PRINCIPLES OF SECURE DESIGN

The following principles are adhered to in DFCS Messaging implementation of the measures proposed above:

Open Design - Only the keys need to remain private for the designed protocol to be effective in providing additional security to SMS messages

Least common mechanism - The GSM modem and network will not be depended on to provide all message security. Rather, additional security is provided at the application level on the phone and the GSM modem is only needed for transmitting protected data.

Economy of Mechanism - All the additional security is handled by the use of one application, i.e. the software designed and implemented for this project, which means that analysis of the security can be restricted to an analysis of the functionality implemented by the application. The security of the GSM network and the phone is important for analysis of messaging security, in general, but such analysis is outside the scope of this project and is irrelevant because the software will not depend on any particular security provided phone's file system, files, or service provider.
The authentication protocol is also relatively simple, as it is common practice in setting up VPN connections using the TCP/IP protocol.

Separation of duty - In order to successfully send secure messages, multiple conditions must be met. Namely, both users must be using trusted phone numbers, and know the passphrase associated with the user.

Psychological acceptability - Although adding security to text messages using an application may put an extra burden on the user, nearly all the details of the protection can be hidden to the end user. The user must simply enter a passphrase to the phone to ensure that only the appropriate user is able to send and read secured messages on his behalf. Although an additional step of accepting the request to send a text message is required, the fact that additional security was decided on for sending the message means it is appropriate for the user to need to spend slightly more time and effort with respect to messaging to provide the desired security.

Defense in depth - the protection added by the proposed software is layered on top of any additional data protection provided by the phone and the cellular service provider. Messages are protected, and users are authenticated at the application layer, and ciphertext is still protected by any

Question Assumptions

Assumptions are stated and justified earlier in the paper. Potential attacks and breaches of the security are discussed as well, so the dangers and inherent flaws of the added security can be realistically analyzed and defended against with future work.

XII. CONCLUSION

As cell phone use grows, demand for security associated with cell phone use and applications also increases. Text messaging via SMS messages is a very common communication method with cell phone users, and sometimes confidential information is transmitted via SMS messages.

Standard SMS messaging is insecure. Although it is quick and easy to use, it is not safe to send messages that require user authentication and secrecy. We have been able to develop an application that provided both these services using a secure authentication handshake protocol, perfect forward secrecy using ephemeral Diffie-Hellman key exchange and encrypting the messages with AES encryption. This adds a few more steps that the user has to perform in sending an SMS message, but adds a layer of security, making it harder for an attacker to gain access to private conversations.

REFERENCES

[1] Appgate Network Security. "One-time passwords via SMS Secure Strong Authentication," Appgate Network Security, http://www.appgate.com/index/products/agss/OTP_eng.pdf
[2] Brick House Security, "Cell Phone Spy: Deleted Texts/Data Extractor." [Product page] BrickHouseSecurity.com 2009. (Available October 27, 2009) http://www.brickhousesecurity.com/cellphone-spy-simcardreader.html
[3] Spyandroid.com, "Spy Android SMS," [Software download]. (Available October 26, 2009) http://www.savesmsmessages.com
[4] Jaloop Enterprises Corp. "SMS Spy Phone Package." [Product page]. (Available October 26, 2009) http://www.trusters.com/t_sms_spy_package1.html
[5] Kelly Jackson Higgins. "Encrypted GSM Voice Calls Hacked in Minutes," 2008, DarkReading. [Article]. (Available Octover 27, 2009) http://www.darkreading.com/security/encryption/showArticle.jhtml?articleID=211201467
[6] CryptWare advanced Software. "Message in a Bottle – Elliptic Curve Cryptography for SMS," Ugo Chirico 2009. [Cell phone software solutions]. (Available Octover27, 2009) http://www.ugosweb.com/miabo/index.aspx
[7] "CryptoSMS – triple encrypted SMS" http://www.cryptosms.com/
[8] CryptoPhone. "GSMK CryptoPhone 300" [Technical Specifications].
[9] Mark Stamp. *Information Security, Principles and Practice*. [Textbook]. 2006, John Wiley & Sons, Inc., Hoboken, New Jersey.
[10] Ross Anderson. *Security Engineering.* (Second ed.) [Textbook]. 2008, Wiley Publishing, Inc., Indianapolis.