Cryptography

<Students' names redacted>

CS&E 551

Introduction

Cryptography is the practice of studying and hiding information. The process of transforming information using an algorithm is called encryption – a branch of cryptography. Protecting information is an important idea in today's expanding virtual world. People want to know that their sensitive information is safe and several methods are in place to insure that. One way to protect data is with an mobile encryption application to protect text messages from being read by friends or family. This type of application would provide a secondary source of protection if a password or lock screen were to be breached. Before this application was created, several challenges had to be overcome to provide a secure way to send data between two people [1].

History

The history of cryptography began as early as the 36th century with the Egyptian writing system of hieroglyphs. Hieroglyphs were a combination of single-consonant characters known as phonetic glyphs; logographs which were the essential units of the words; and determinatives which reduced the meanings of the glyphs and logographs. Hieroglyphs were used to communicate between individuals to share cooking recipes and religious documents. While hieroglyphs were not a true form of cryptography, they were used as a way to mystify, intrigue, and amuse other literate by-passers [2].

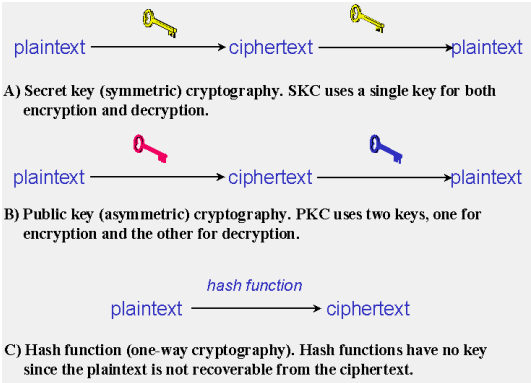
As the centuries passed, basic forms of cryptography appeared from around the world. Between 600-500, Hebrew scholars created a simple substitution method with their alphabet. In the early 100s CE, Romans expanded on the Hebrews' substitution method with the Caesar cipher which shifted the alphabet by a fixed number of spaces [3].

Encryption has also played a large role in world politics. During World War I, Arthur Zimmermann, the Foreign Secretary of the German Empire, sent a coded telegram to the German ambassador that was stationed in Washington, D.C. The German ambassador forwarded the message to the German ambassador in Mexico in an attempt to convince Mexico to attack the United States. Mexico declined to attack the United States even after being promised land that they lost in earlier wars against the United States. This message was intercepted by the British and decoded by their cryptography effort – Room 40. The decoded message was then sent to the United States President where it was passed to the presses. This caused a public outrage and was one of the factors that pushed neutral United States into World War I [4].

Up until the 1970s, encryption was not a publicly used idea. Even the government did not have a standard for encryption. Several failed attempts were made to create a standard but nothing was accepted until 1974 when IBM submitted a symmetric-key algorithm that seemed acceptable. The chosen name, Data Encryption Standard (DES), was published for public comments in March 1975. There were several pieces of criticism for the short 56-bit key and the mysterious "S-boxes" added from the National Security Agency (NSA). Critics worried that secretly adding new technology would severely weaken the algorithm since only the NSA knew how it worked. With a government-wide, public standard established, amateur programmers attempted to "crack" the algorithm. This united many people and the field of cryptanalysis grew steadily. DES was eventually replaced by Advanced Encryption Standard (AES) but any new encryption algorithm is still compared to DES [5].

Types of Cryptography

There are three basic types of cryptographic algorithms: symmetric, asymmetric, and hash function. Symmetric cryptography uses the same key for both encryption and decryption. For this reason it is often called secret key cryptography. Asymmetric encryption uses a public key for encryption and a private key for decryption. Thus, it is also known as public key cryptography. Hash function cryptography does not use a key because once information is encrypted, it cannot be decrypted. Each of these three types is depicted in the diagram below [6].



A) Secret key (symmetric) cryptography. SKC uses a single key for both encryption and decryption.

B) Public key (asymmetric) cryptography. PKC uses two keys, one for encryption and the other for decryption.

C) Hash function (one-way cryptography). Hash functions have no key since the plaintext is not recoverable from the ciphertext.

In symmetric cryptography, the sender must use the key to encrypt the plaintext. They must then send the cipher text to the receiver and the receiver must use the same key to decrypt the ciphertext. One of the biggest difficulties with symmetric cryptography is exchanging the key without revealing it to outside sources. On the other hand, it is very efficient. Examples of symmetric algorithms include Data Encryption Algorithm (DES), Triple Data Encryption Algorithm (Triple-DES), Advanced Encryption Standard (AES). Symmetric algorithms can be categorized as block ciphers or stream ciphers. Stream ciphers encrypt each bit of the plaintext into a bit of ciphertext one at a time. Block ciphers on the other hand, encrypt a block of the plaintext at a time, using the same key on each block [6].

Two types of stream ciphers are self-synchronizing stream ciphers and synchronous stream ciphers. A self-synchronizing stream cipher calculates each bit of the key as a function of the previous n bits in the key. Decryption is able to stay synchronized with encryption just by knowing how far into the n-bit key it is. Synchronous stream ciphers generate a string of random digits independently of the plaintext and ciphertext. This is then combined with the plaintext for encryption or with the ciphertext for decryption [6].

There are several modes of operation for block ciphers. Four of the more important modes will be discussed. Electronic Codebook (ECB) is one of the most basic modes. In this mode, the message is separated into blocks and each block is encrypted by itself. Identical plaintext blocks then generate identical ciphetext blocks. Cipher-Block Chaining (CBC) mode XORs each plaintext block with the previous ciphertext block before encrypting it. An initialization vector must be used to encrypt the first plaintext block. Cipher Feedback (CFB) mode essentially creates a self-synchronizing stream cipher out of a block cipher. It is almost identical to Cipher-Block Chaining performed in reverse. Finally, Output Feedback (OFB) mode generates a synchronous stream cipher from a block cipher. It generates key blocks which are XORed with the plaintext blocks in order to generate the cipher text [6].

With asymmetric cryptography, two different, but related, keys are used for encryption and decryption. This takes away the need to secretly exchange the key between the sender and receiver of the message. When using an asymmetric algorithm, the intended recipient of the message generates both keys. The intended recipient then publishes one of the keys (known as the public key) which the sender uses to encrypt the message. Upon receiving the message, the recipient decrypts the message with his private key. Public key algorithms depend on one-way functions. A one way function is a function that is relatively easy to compute, while its inverse is relatively difficult to compute. Examples include multiplications vs. factorization and exponentiation vs. logarithms. One of the most popular asymmetric algorithms is RSA. They are also used with digital signatures to protect the authenticity of a message. A digital signature is created using the private key and verified using the public key [6].

The last of the three types of cryptography is a hash function. Hash functions are a means of one way encryption. They take a block of data, called the message, and return a fixed size bit stream, called the hash value or digest. The four main properties of a hash function are that it's easy to compute the hash value for any given message, that it isn't feasible to find a message given its hash, that it isn't feasible to modify a message without the hash being changed, and that it isn't feasible to find two messages with the same hash value. Hash functions are commonly used by operating systems to encrypt passwords and to provide a digital fingerprint of a file's contents to ensure that it has not been altered [6].

Problems Faced

Encryption Method

We really did not have many problems, but this section covers what problems we did have and the solutions to those problems. To outline our problems we had a few encryption issues, problems getting the text message from the inbox to the application and also we'd like to talk about installing and using windows 7 phone development tools which you is typically a good source of problems, especially for firth time developers on a platform.

We started our project before we had any lectures on encryption in class and although we have all touched encryption in other classes we hadn't thought of a simple way to encrypt our message. We started with an encryption method

we came up with which encrypted by taking the ASCII integer value and adding the pin number then taking that sum and mod that value by 127 to come up with a character that is printable and within the 0 to 127 ASCII range. From there it was decrypted by going in the opposite direction, so subtracting the Pin as an integer mod by 127 and then adding 127 because you want a positive value and subtracting the Pin will always get you a negative number. Unfortunately for some reason part of the string was being truncated and after some investigation and debugging we were unable to find the source of the problem which was only an occasional occurrence. Fortunately after we had our lecture on Encryption and talk about the XOR operation it was obvious we could use our pin number and the XOR operation to encrypt/decrypt our messages.

Copy SMS to App

Our biggest problem was figuring out a way to copy the SMS message to our application. The problem is developer tools have no access to text messages and with brainstorming and searching the internet we had several idea, some of which were better than others but proved to be impossible to implement due to restrictions. The three main possibilities were inserting a link into the message, host our own messaging server, or use copy/paste functionality already provided by the phone. Our most preferred method would have been inserting a link into the message. Simply put we would insert a link into the message which would link to the location of the application on the phone and we would provide a query string which would contain the encrypted message. The next method which we hardly even thought about was trying to set up our own messaging server which the application could connect to and download/upload messages for other users. Obviously this is a huge undertaking, especially for a class like this so we were lacking the time resources and not to mention having a server or computer we could host this from. The last method was to use the Copy/Paste functionality of the phone and the problem with this solution was that it would require more user interaction than would likely be desired by the user. Additionally without an updated phone the Copy/Paste functionality would not be available and would be a necessity for our application.

So the link was the most ideal method in our opinion because of the increased usability of our application and relatively simple implementation. The problem was that the Windows 7 Phone development tools do not currently allow it. The reason is because of security issues which with the use of a link someone could install something malicious on the phone. However other phone OS do provide this in their development tools and for Windows 7 to compete better with the other phones they too will hopefully open up this functionality in the future. So that being said we obviously chose to just use Copy and Paste, which works just fine in our app even with a slight loss of usability.

Windows 7 Phone Development Setup

We did not really have problems with the windows 7 phone development tools but it could be a big source of problems so we wanted to give credit to Microsoft for making it so simple. All you have to do to develop a Windows 7 Phone application is follow these steps: install visual studio 2010, install the development tools for windows 7 phone, create a project for a windows 7 phone (which is an actual option when you go through the create new project wizard) , write the code and run the project which will start up your application in the emulator. Going into further depth the new project wizard and drag and drop functionality along with generating code for events saves a lot of time in coding making it almost unnecessary to know how to do those things and allows you to focus on the important 20% of the code. It was all very simple and even with inexperienced .Net users we were able to focus mainly on functionality rather than designing the UI and event handling.

How the Code Works

To start off, as was mentioned the Visual Studio Express for Windows 7 was used for the majority of the work. This IDE has lots of built in tools to generate basic code. This made creating the user interface extremely intuitive. The other tool we found that was helpful for making the app interface appear professional was Expression Studio 4. We learned of this tool later in the development process so its uses were limited but it allowed finishing touches to be made to make the product feel more polished.

One important feature that we took advantage of in Visual Studio was its drag and drop interface. Visual Studio's interface for developing phone apps was set up to have the code portion on the right, a representation of the current page of the app on a simulated phone on the left. To the left of the phone was a toolbox that allowed for object to be selected and dropped onto the simulated phone. This would automatically generate code. This allowed us to spend the majority of our time working on the actual app instead of spending our time tweaking minor interface options.

The code for out app was spread out over three categories, the App, MainPage, and the PinInput page. These pages were all linked through the navigation tools that will be covered shortly. Each code section contained both a xaml

file and a linked cs file. This set up allows for the application code and the user interface code to be separated and viewed as individual entities. The xaml files were where the user interface code was held. This is the portion of the code that took advantage of the aforementioned drag and drop tools.

The main file for out project was our pinInput file. This file had four methods that completed the majority of the application's work. The first was the button clicked method. this method utilized the two Navigation classes we took advantage of to move around our app and access data. These two navigation classes were the Navigation Service class and the Navigation Context class. These classes allowed us to easily move between pages in our application and access various data throughout it. The service class allowed us to link through parts of our pages. The navigation context class used query strings to identify different fields of data to access from throughout the many parts of the program.

```
This.NavigationService.Navigate(new
Uri("/MainPage.xaml?Text=" + text,
UriKind.Relative));
```

In the above code sample the navigation service class is used to bring the appropriate text from this pinInput file and put it in the correct place on the main page.

```
this.NavigationContext.QueryString["Mode"] ==
"Encrypt"
```

The above code sample uses the navigation context class to do a check of the mode that was stored to determine if the user intended to decrypt or encrypt the message. It also takes advantage of query strings. The query string technique was a useful way to find data we needed despite not having it in the current context.

The pininput file also contained the work that was done for the en/decryption methods. The technique for these methods has already been covered but it should be noted that both methods used the queryString to obtain the text data that was needed. Finally a fourth method was used to monitor any text box changes from the user entering the pin. This method would keep the page up-to-date on any changes.

One last major point of the code was the sms api we used. The windows phone development provides many features called launchers and tasks that enable easy access to common phone features including calls, calendar, texting, and taking pictures. These launchers create a secure way for the user to switch from third party apps to their personal data without the app needing to know any private information. They are essentially black boxes that developers can use to allow their app to do more without being invasive. For us the SMSCompose task was an extremely beneficial tool to bring our final decrypted message right into a text message. The task takes the information from our main text box and places it into a text that the user can then choose who to send it to.

How to Use the Application

The user interface for the SMS Encryption application is designed to be minimal and straightforward. Due to the fact that this application is essentially a text message utility, it is important that the user be able to navigate and be presented with as few menus and options as possible. This way, the user can experience the rapid pace of communication offered by SMS text messaging. In fact, the developers' intention is for this application to become the user's primary means of sending and receiving text messages by providing an extra layer of security. The following is a description of how to use the SMS Encryption application.

To begin, the user must install the application on a Windows Phone 7 handset. This can be accomplished in two ways: First, as a developer, it is possible to install the application using the Zune client software package on a PC and connecting it to the phone. Second, assuming the application is available on the Windows Phone Marketplace, the application can be installed by using the phone itself via the 'Apps' section of the Marketplace. This is the primary method of installation for normal users and/or customers. Once it is installed, the user can launch the application by tapping its icon on the phone's start screen.

Upon launching, the user is presented with the application's main page, which consists of a text area and three buttons. By tapping anywhere within the text area, the user is presented with the phone's SIP (Soft Input Panel) - a virtual keyboard. The user can enter a text message into the text area using this keyboard; when finished composing, the user can dismiss the keyboard by tapping anywhere *outside* of the text area. Once the keyboard is dismissed, the user has the option to press the 'encrypt' button to encrypt the message. Upon pressing this button, the user is prompted to input a four-digit PIN number to act as an encryption key. In other words, the application uses the PIN number to scramble the characters of the text message into a new indistinguishable string (the same PIN number is used to *decrypt* the message and restore it to its original value). Subsequently, after the user inputs his or her PIN

number, he or she is taken to the main page and the newly encrypted message is displayed in the text area.

To send the encrypted message as an SMS, the user simply taps the 'to SMS' button on the main page. When pressed, this button launches the phone's native SMS application and the encrypted message is inserted into a new SMS body. At this point, the user only needs to select a recipient for the message and press the 'send' button to initiate an SMS transmission. Once this process completes, the selected recipient will receive the encrypted message on his or her phone.

In order for the recipient to decrypt the encrypted message, he or she must also have the SMS Encryption application installed on his or her handset. From the default SMS application, the user must copy the encrypted message and paste it into the text area of the SMS Encryption application. Then, the recipient can press the 'decrypt' button on the main page. At this point, the recipient must enter the same PIN number used by the sender in order to successfully decrypt the message. When the correct PIN number is entered, the encrypted message will be restored to its original readable state in the text area on the main page. As this case illustrates, it is important that both the sender and recipient find a way to communicate the PIN number in a secure manner. This is a task for the user(s) to accomplish independently.

In summary, the SMS Encryption application provides the following functionality:

- Intuitive interface
- Text encryption (symmetric key)
- Text decryption (symmetric key)
- Transmission of encrypted text as SMS messages

Conclusion

As the concept of encryption expanded onto computers and eventually mobile devices, newer standards and ideas were needed to secure the safety and privacy of information. Mobile platforms already provide tools for protecting information by restricting access to several parts of the mobile operation system. Using a secondary system to protect the owner's private information - including text messages, emails, and pictures – only adds further security to a rapidly growing market.

References

[1] "Cryptography | Define Cryptography at Dictionary.com." *Dictionary.com | Free Online Dictionary for English Definitions*. Web. 13 Mar. 2011. <http://dictionary.reference.com/browse/cryptography>.

[2] Krebs, Robert E., and Carolyn A. Krebs. *Groundbreaking Scientific Experiments, Inventions, and Discoveries of the Ancient World*. Westport, CT: Greenwood, 2003. Print.

[3] "Caesar's Method -- from Wolfram MathWorld." *Wolfram MathWorld: The Web's Most Extensive Mathematics Resource*. Web. 13 Mar. 2011. <http://mathworld.wolfram.com/CaesarsMethod.html>.

[4] "Zimmermann Telegram." *SimonSingh.net*. Web. 13 Mar. 2011. <http://www.simonsingh.com/Zimmermann_Telegram.html>.

[5] "RSA Laboratories - 3.2.2 Has DES Been Broken?" *RSA, The Security Division of EMC: Security Solutions for Business Acceleration*. Web. 13 Mar. 2011. <http://www.rsa.com/rsalabs/node.asp?id=2227>.

[6] Kessler, Gary C. "An Overview of Cryptography." *GaryKessler.net*. 03 Feb. 2011. Web. 03 Mar. 2011. <http://www.garykessler.net/library/crypto.html>.