An Innovative Approach Towards Applying Chaum Mixing to SMS

Matthew Patrick Rothmeyer
University of Arkansas, Fayetteville

An Innovative Approach Towards Applying Chaum Mixing to SMS

An Innovative Approach Towards Applying Chaum Mixing to SMS

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science in Computer Engineering

by

Matthew Rothmeyer
University of Arkansas
Bachelor of Science in Computer Engineering, 2012

May 2014
University of Arkansas

This thesis is approved for recommendation to the Graduate Council.

Dale R. Thompson, Ph.D., P.E.
Thesis Director

Jia Di, Ph.D.
Committee Member

Brajendra Panda, Ph.D.
Committee Member

ABSTRACT

Currently there are few user-friendly applications for anonymous communication across multiple platforms, leaving data that is often both personal and private vulnerable to malicious activity. Mobile devices such as smartphones are prime candidates for such an application as they are pervasive and have standardized communication protocols. Through the application of mixing techniques, these devices can provide anonymity for groups of individuals numbering 30 to 40 members. In this work, a Chaum mix inspired, smartphone based network that uses the Short Message Service (SMS) is described first in theory and then in implementation. This system leverages both techniques used by current anonymity networks as well as knowledge gained from current and past research to make messages private and untraceable. The work addresses previously published attacks to anonymous systems through current and innovative mitigation techniques.

ACKNOWLEDGEMENTS

I thank Dr. Dale Thompson, Dr. Brajendra Panda, Dr. Jia Di, and all of the faculty who helped guide my development as a scientist and engineer.

I also thank my family, my friends, and my fiancée, without which I would have likely finished this thesis a year earlier, but would have enjoyed it far less.

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION

1.1 Motivation

While difficult in the past, the process of tracking, cataloging, and categorizing information about an individual in the present day has become relatively simple by comparison. A majority of present communication is accomplished through some sort of digital medium that, more often than not, leaves remnants after the logical lifetime of the initial interaction. Even communication that does not exist in electronic form often has a record of that kind associated with it. These remnants or records can be considered an electronic footprint; defined as data that exists as the result of some interaction with an electronic system, either direct or indirect.

The unfortunate reality is that most individuals are at best only partially aware of one's electronic footprint and at worst are unaware that this information exists at all. Of even greater importance is the understanding that an electronic footprint can often be used to gather and infer information about an individual that would be considered private or at least not germane to the intent of the initial interaction (inferring listening preferences based upon music of purchases for example). In some cases, these inferences are harmless and could be considered within the ethical bounds of a particular entity (a music application suggesting new artists). Problems arise however when these inferences are not harmless or would not be considered to lie within the ethical boundaries of an entity. This is often the case when information is sold or acquired without the user's express permission.

The core problem is that in most cases a user is not able to communicate or interact electronically in an anonymous manner. With anonymity, even if that user's communication becomes compromised there is no way to tie any inferences made using that data to an individual. This is not to say that there are not malicious uses for compromised data (say credit card information), but simply that anonymous communication increases the difficulty of using mundane information in a harmful way. Sadly, most solutions for anonymity are either unimplemented or too difficult for someone without sufficient technical experience to use. As a result, data that most individuals would consider private is vulnerable and can lead to significant damage if misused.

1.2 Objective

In this work, a Chaum mix inspired, smartphone based network that uses the Short Message Service (SMS) is developed. This system solves a portion of the anonymity problem by providing an easy to use framework for small groups numbering 30 to 40 on a pervasive platform through a common communication medium.

1.3 Approach

The process of development can be broken up into three distinct categories: theory, code development, and testing. The first category involves developing the theory behind the application. In order to design the proposed framework it was necessary to either develop a new method of making communication anonymous or repurpose an existing methodology for a new platform. In the case of this particular project, the latter option was chosen. The Chaum Mix was selected as the basis for the protocol as it was initially written for electronic mail and in theory should work for any communication protocol functions by sending distinct messages. It was then determined how key properties of Chaum Mixes might be applied to SMS. This involved taking into account the limitations of both the SMS service and mobile devices on which the protocol is running. Finally, problems faced by both the Chaum Mix and the SMS Mix, as well as how they might be mitigated, were addressed using current and past research.

Once theory detailing how the SMS Chaum Mix would operate was made concrete the development of code began. The initial implementation of the algorithm was done only in Java using a simple message passing framework in place of SMS. This framework used the same interface as SMS allowing for rigorous unit testing of the mix network without requiring use of mobile devices. Once the algorithm was tested the code was moved to the Android mobile operating system.

Finally, the application was fully tested. This is in addition to the initial testing of the protocol on personal computers. Testing took place on multiple types of smartphones across several versions of the operating system to verify compatibility with both the software and the hardware aspects of the phone.

1.4 Organization of this Thesis

The rest of the work is organized as follows. Chapter 2 covers background information pertaining to the SMS Chaum Mix. This involves discussing key concepts relating to the need for anonymity as well as how the Chaum Mix operates. It also discusses current and past approaches to anonymity both Mix based and otherwise. Implementation of the proposed system is detailed in chapter 3. Starting with an overview of the design at a high level of abstraction this chapter is divided into sections detailing the theory, code design, and testing that went into the system. Chapter 4 discusses the results of the testing described in Chapter 3 as well as what these results mean and how they were analyzed. A summary description including how this work contributes to the community and what future work might take place is discussed in Chapter 5.

2. BACKGROUND

2.1 Anonymity and Secrecy

In order to grasp the nature of the problem addressed by this work and understand the work others have put forward, it is important to have a firm understanding of anonymity and secrecy are well as the terms often used in conjunction with them. While there are many possible ways one could go about this, Pitzman et al. developed a detailed set of terminology and definitions to standardize the way we speak about such issues. In [1], a system is the overarching area in which communication takes place. It is a connection of interacting devices in a communication network. An area observed by an adversary is known as that adversary's domain. Anonymity is "the state of being not identifiable within a set of subjects" with that set of subjects being termed the "anonymity set." Secrecy, when related to messaging protocols such as ours, involves providing a system "where the meaning of the message is concealed by cipher, code, etc., although its existence is not hidden." [2].

In the case of this work, the overarching goal is to preserve the secrecy of a message and at the same time prevent a particular sender and receiver from being associated with the sending of messages. Preventing this association is defined as providing unlinkability. Unlinkability does not assume that an adversary cannot watch the network (known as unobservability), but simply that even if one could observe that it would be inordinately difficult to determine who was communicating with whom. As a final point, it is important to discuss the difference between what Pfitzmann define as anonymity and pseudo anonymity or pseudonymity. Pseudonymity is "the use of pseudonyms as IDs" and results in the actions of users being linked with some identity other than their own (the pseudonym), while anonymity is the state in which an actor cannot be attached to any part of the anonymity set.

A large body of research already exists in the area of anonymity and secrecy in communication systems. This research often takes the form of protocols and rule sets applied to specific classes or modes of communication. Many of these protocols, taking their cues from Chaum's initial research into mixes [3], are formed of loosely connected computational nodes capable of interacting in a fashion similar to a peer-to-peer network. These nodes are usually distributed and rely on sets of volunteered hardware instead of some overarching organization to provide the network backbone.

While originally directed at electronic mailing networks, this form of anonymity has seen applications in many areas ranging from hiding web traffic in the onion router [4] to Telephony [5]. An understanding of Chaum Mixing along with resulting research is essential in identifying key characteristics necessary for developing anonymous systems of the same class. With an understanding of Chaum Mixing, one could deduce (among other things) that almost any network of devices capable of meeting the constraints defined above could achieve anonymity through an application of the concepts proposed by Chaum and the research inspired by them.

2.1.1 Current and Past Approaches to Anonymity

Anonymity networks see application across a wide range of technology domains. Onion Routing [6] was developed as a method for anonymous communication via TCP/IP. The onion router finds its foundation in a set of global volunteer nodes, each obfuscating (or rendering obscure) any traffic that passes through it. Anonymity is achieved by directing traffic through several such nodes in sequence. In addition to this obfuscation, data to be sent is encrypted in a layered structure with each layer corresponding to a hop between nodes. When a node receives data it removes the outer layer within which it finds routing information and an encrypted payload, and then forwards that data to the next node. This process continues until the data reaches the last node. At this point data is decrypted and is sent as plaintext (from the perspective of the network) from the last node to the intended recipient. An example of Onion Routing with letters is shown in Figure 1.

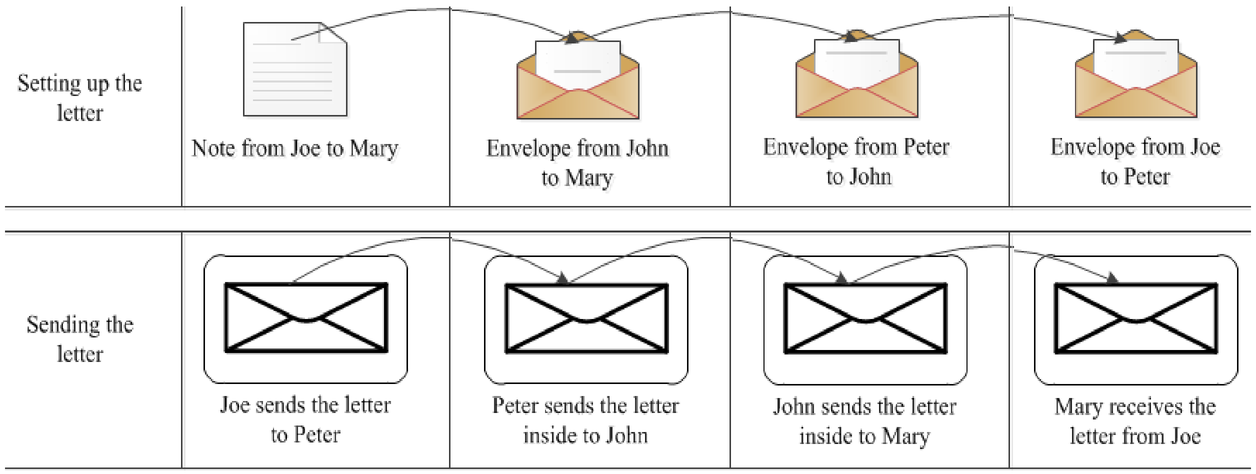

Setting up the letter
Note from Joe to Mary
Envelope from John to Mary
Envelope from Peter to John
Envelope from Joe to Peter
Sending the letter
Joe sends the letter to Peter
Peter sends the letter inside to John
John sends the letter inside to Mary
Mary receives the letter from Joe

Figure 1: Onion routing with envelopes

Freenet [7] is a system designed for anonymous data sharing across distributed hardware. This distributed hardware is mostly composed of personal computers belonging to users who, in a sense, donate unused storage on their machines. Files are identified using a two-part key derived from a hash of the file. The first part of the key is public is used for identification while the second half, used to sign the file, allows for integrity checking. The public key and file are then posted to the network where the file will be distributed through the system with files determined to be similar located within close proximity of one another.

Crowds [8], a system designed with the express purpose of browsing web pages anonymously, uses a similar volunteer framework as Freenet. A user, known as a jondo (John Doe), exists in a group of users called a crowd. Users are assigned to a group and informed of new members through an administrative daemon known as a blender. Web interaction is made anonymous by routing requests through a random route of other jondos in the network. In route negotiation decisions such as which node to add, as well as the length of the route, are established using a biased coin flip. Responses are returned to the initiator by traversing the route in reverse from server to client. At any point a jondo only knows what node exists before and after it in a route.

Tarzan [9] is a system that combines properties of Onion routing and Crowds while adding new ideas. Like onion routing, Tarzan is designed to make network traffic anonymous using encrypted routes or circuits, though this traffic uses the UDP transport protocol instead of TCP. In Tarzan, all users send data for all other users (as in crowds) and users discover other users through peer-to-peer communication with neighboring clients. Unlike other designs, Tarzan also introduces measures to combat active observation, specifically by introducing false traffic between nodes.

Mixminion [10] is a system that builds directly off the Chaum Mix by providing the ability to make email anonymous. Also known as a type III remailer, it is the successor to the Cypherpunk and Mixmaster remailers (Type I and Type II). Mixminion routes messages through a series of email servers to hide the originating source of an email. Using layered encryption, Mixminion assures that any message entering a server cannot be correlated to a message leaving that server (as inbound messages are decrypted before being sent). Mixminion improves upon the Type I and II remailers by simplifying the process of obtaining information about the available remailing servers as well as covering security flaws exposed in Cypherpunk and Mixmaster.

2.1.2 The Chaum Mix

Each of the above systems draws inspiration from Chaum's initial work on mix networks. As such, it is important to understand the Chaum Mix at a high level of abstraction at the very least. The mix network is a set of steps and algorithms that, when used together, allow an individual to send messages or information without revealing the identity of the sender. Chaum defines an environment where interaction exists between what are referred to as senders and mixes. A sender is some user or device that introduces traffic into the system, while a mix is a node that processes an item of traffic or mail. Mail introduced into the system by a sender S directed to a recipient X is first padded with random bits and encrypted using a public key K belonging to X. Let M be the message to be sent, R be a set of random bits added to prevent any sending of the same message text from encrypting to the same value, and C(x,y) be the encryption function where x is the key and y is the plaintext. The encrypted message EM is then EM = C(k,{M,R}). Encryption prevents any intermediate party from reading the data within, while random padding prevents an observer from reverse engineering the message data through repeated encryptions with the recipient public key. This random padding is also useful as it results in different generated encryption values even if the message content is the same.

At this point, the number of mixes the message will traverse as well as the route that message should take are chosen. The Chaum Mix assumes that any sender can access a list of available mixes with corresponding public keys. Once the route is decided, the sender will then encrypt the message in layers using keys associated with each selected mix. Each layer contains three parts: a payload composed of all of the encrypted layers below it, an address for where the message should be sent next, and some random bits. Layering begins with the last mix a message will reach and ends with the first. Using the same terminology as before and assuming 3 mixes are used in numerical order, let K_i be the key of mix i, R_i be a set of random bits for a mix, and A_i being the address of mix i. The payload sent by S, called P is then constructed as:

$$P_1 = C(K_3, \{E_M, R_3, A_X\}) \tag{1}$$

$$P_2 = C(K_2, \{P_1, R_2, A_3\}) \tag{2}$$

$$P_{Final} = C(K_1, \{P_2, R_1, A_2\}) \tag{3}$$

When the message arrives at a mix said mix removes the outer layer using its private key, retrieves the address data, and forwards the payload to the next mix while discarding the random bits. This process repeats until the message reaches the final mix. This last mix sends its data to the intended recipient who can decrypt the message. The decryption process D_i across a route where mix i decrypts is as such:

$$D_1(P_{Fin}) = D_1(C(K_1, \{P_2, R_1, A_2\})) = P_2, R_1, A_2 \tag{4}$$

$$D_2(P_2) = D_2(C(K_2, \{P_1, R_2, A_3\})) = P_1, R_2, A_3 \tag{5}$$

$$D_3(P_1) = D_3(C(K_3, \{E_M, R_3, A_X\})) = E_M, R_3, A_X \tag{6}$$

Aside from the encryption scheme, Chaum also drew attention to the importance of how and when data was sent from a mix. One significant problem in maintaining anonymity is preventing observable data, such as arrival and departure times, from inadvertently violating the anonymity of a user. In order to prevent this scenario several parameters such as size, sending frequency, and organization must be made standard. To accomplish this each mix sends received data using fixed size, lexically organized batches. Each batch is sent to at least any user or mix that would be receiving data within that batch. It is left up to the next recipient to select appropriate messages from the batch and discard the rest. In order to prevent an adversary from correlating the number of input and output messages, each mix should include randomly addressed dummy messages in order force all message batches to meet some minimum required size. A mix must also prevent duplicate or repeated messages. A repeat would allow an attacker to correlate the repeated input with the repeated output (as those messages would be the only ones that were both sent to and sent from that mix) and thus violate anonymity for that message in the mix. A diagram of a single node mix network is shown in Figure 2.

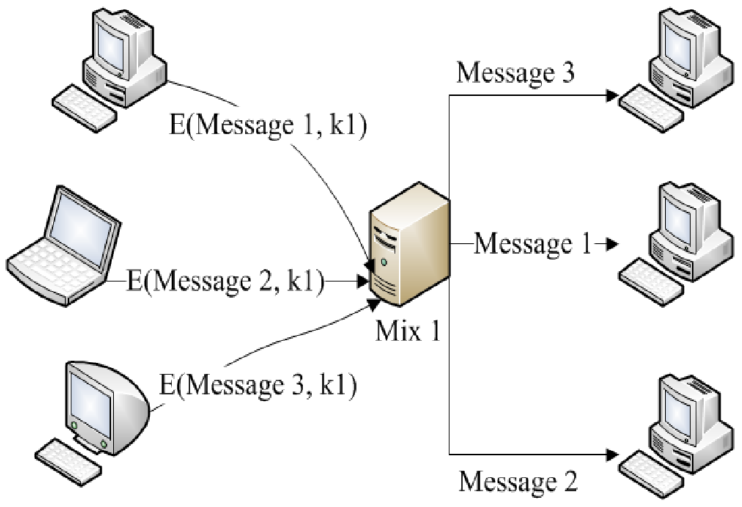


Figure 2: A single node Chaum Mix

2.2 Related Work

Since the initial proposal of the Chaum mix much research has been done in the domain of anonymous communications. In any field of research it is important to understand what has been accomplished by those who have come before in order to better grasp the field in which one is working, to learn from the mistakes and accomplishments of others, and to prevent the same work from being done more than once. The following section is a summary of the research and accomplishments of others that acts as a foundation of this work.

2.2.1 Classification of Networks

In their work, Edman and Yener [10] give an overview of much of the current and past anonymity research. In this work, they note that while there are several ways to classify an anonymity network each falls into one of two categories based on latency, specifically high- and low-latency networks. Latency is an important characteristic in anonymity networks as it often is the deciding factor in what steps can be taken to achieve anonymity with regards to time. A high-latency network is one in which interaction between actors can tolerate delays of up to several hours or more. Systems that fall into this category are often messaging systems such as email. A low-latency system is one in which delays are far less tolerable and in some cases prevent the system from functioning properly. Low-latency systems are often fall into the domain of interactive applications such as voice or video communication or data transfer in which the size of the data to be transferred is large in comparison to the size of the data within a packet. High-latency systems are also referred to as message based systems while low-latency systems are known as connection based. One useful aspect of the above system of classification is that it aligns with the types of attacks users might face. An adversary wishing to compromise a high-latency system would take a different approach than an adversary whose goal was to attack a low latency system.

2.2.2 Attacks faced by Anonymity Networks

In the years since Chaum's initial work, many attacks against such systems, along with methods for countering such attacks, have emerged. An important part of designing any anonymous system is understanding how to apply past research to current work, in this case methods for countering adversaries. In order to appreciate and understand those methodologies, one must first understand how an attack functions. As such, it is important to examine the threats that both low- and high-latency systems encounter, and determine which threats need to be mitigated.

Edman and Yener [10] as well as Marques and Zuquete [11] have outlined several of the most common methodologies of attack in their work. These include the blending attack, the predecessor attack, the intersection attack, the timing attack, and the Sybil attack. Each attack is discussed below.

Blending Attacks

One of the most common classifications of attack (referred to by [12, 13, 14]) is known as the blending attack. Within this classification there exist two forms of attack, the (n-1) attack and the trickle attack, which are most often directed against low-latency systems (though they are not exclusive to these). Each of these attacks builds on the knowledge that buffer size in any messaging system is made finite either by the physical constraints of the system (running out of memory) or the constraints of the protocol (data can only sit in the buffer for so long). Regardless of why the buffer may be finite, an adversary knows that some sort of flushing algorithm must be used to clear items from the buffer to make room and to facilitate delivery.

The (n-1) attack is specific to the classic mix proposed by Chaum. In this case, a mix will flush or send messages when some threshold for a number of stored messages has been met. In this attack, an adversary will select a legitimate message M to track and will then intercept and delay said message from its next hop. Then, the attacker will attempt to clean out the next mix in M’s route by flooding that mix with fake messages until the adversary observes a flush. Once the flush occurs, the attacker will send M along with more fake messages (n-1 fake messages, where n is the size of the buffer) until the mix flushes a second time. The attacker then intercepts this second batch and eliminates or removes the dummy messages it provided. What is left is at worst a small set of messages that could be M (assuming some messages were sent from other mixes during the flood), or at best only M. Since anonymity in a Chaum Mix largely comes from an adversary not being able to identify or correlate incoming and outgoing messages, this form of attack eliminates or severely restricts the effectiveness of the mix, especially considering that the process could be repeated on subsequent mixes. In the (n-1) attack it should be noted that an attacker must be able to interact with the network to some degree, specifically to be able to send many messages and to intercept a message M. This can often be accomplished through enrolling oneself as a Mix in the network.

The trickle attack works in a similar way to the (n-1) attack but is directed at mixes whose flushing algorithm sends all messages received within a certain window of time instead of when the buffer is full. In this scenario, the attacker waits for a batch to be sent and then allows only a single message M into the mix. On the next flush, the attacker is guaranteed to see the mix output the decrypted message M as no other messages were able to enter the mix. It should be noted that the trickle attack requires an adversary to have a fair amount of control over the network being observed in order to be able to stop all messages destined for a mix.

Predecessor Attacks

The predecessor attack, first suggested by Reiter et al. [15] and then formalized by Wright et al. [16], can be considered a step up from blending attacks in both complexity and ingenuity. In this attack, a group of adversaries (or a single adversary controlling multiple hosts) joins an anonymous network and attempts to identify the sender in a conversation or connection. In many anonymity frameworks, communication takes place along some negotiated path or route. Often this route is remade on set time periods in order to prevent correlation. As communication continues it becomes increasingly likely that, as time passes, the sender will select an attacker as the node that directly follows it when the path is remade. If the attackers can identify a stream or connection after a path is remade (possibly by some pattern in how data may be sent), they can eventually identify the sender. A predecessor attack can also be conducted by having an adversary as the first node into the network and the last node out of the network. By gathering timing information from the endpoints of the network, a group of attackers can discover which nodes are communicating with each other as timing patterns coming from a sender will often match those on the receiving end.

Intersection Attack

The intersection attack is a passive attack, which has the same goal as a blending attack in that it attempts to correlate a message entering a mix to a message exiting a mix. This form of attack requires that an adversary be able to monitor traffic between devices on an anonymity network, determine an anonymity network's connected users (note that this is different than being able to tell which users are communicating with each other), and to understand the underlying flushing algorithm of that network. To begin, an attacker first selects a user X to track and then observes the mixes said user directly communicates with. By carefully watching for outputs that occur after X has sent a message, an attacker can determine which outputs must have a message that belongs to X and which do not. The next step in the attack uses the knowledge that every message sent by X to a recipient will move through the same set of mixes, a rule that all other messages in a batch are not bound by. By observing many batches known to contain a message from X over a period of time, and taking the intersection of all recipients of that batch, an attacker will eventually be left with only one recipient, which is where the message from X will be heading. Through repetitions of this process, an attacker can determine all communication pairs through any number of mixes.

Timing Attack

The timing attack is a more generalized version of the methodology used by the predecessor attack and only requires that an adversary observe a network instead of actively attacking it. In the timing attack, users monitor communication flows based on message timing. If multiple users are in a network, they can determine if they are on the same line of communication. Though only passive observation is required, timing attacks can be sped up using an active approach Zhu et al. [17] proposed that an active adversary could inject traffic that followed a specified pattern or in some other way to induce a timing delay along a route forming a sort of wave in traffic. This wave would not only be present in the initial node, but would also propagate to other connected nodes in a communication route. By observing traffic and watching for the induced pattern an observer could track connected nodes, even back to the initial sender. It should be noted that this attack is more difficult to execute in networks prone to having delay introduced outside of the attacker's control.

Sybil Attack

The Sybil attack, when applied to a distributed or peer-to-peer environment as described in [18], allows an adversary to gain disproportionate influence over the network. A Sybil attack requires that an adversary be able to create multiple identities or personalities on a network, with each of those personalities following the commands of said adversary. These personalities will then join an anonymity network as separate nodes. To an outside observer each node appears to act as a separate unconnected user, but in reality, they are all secretly controlled by the adversary. Having multiple nodes controlled by one user provides an adversary with capabilities that often violate the expected constraints of the network. Once established these nodes can collude, sharing data with the attacker and performing other kinds of attacks in sequence. The more nodes that can be attributed to a single entity, the more effective the attack is.

2.2.3 Solutions to Attacks on Anonymity

Once one understands the attacks faced by anonymity networks one can begin to examine existing solutions with the goal of developing new ones. In blending attacks, almost all solutions involve removing an attacker's ability to modify or control the inputs of a mix beyond what a single user should be capable of doing.

In order to combat the predecessor attack Wright et al. [19, 20, 21] propose that a sort of entry guard might be used as a countermeasure to the predecessor attack. It was noted that in the case where the first node in a network was trusted, the scenario in which the user was the predecessor to the attacker was not possible. In every case, all an attacker concludes is that the initiator of the communication is one of the entry guards.

Intersection attacks are very difficult to combat. In their survey of anonymous communication systems, Edman et al. [10] were able to find no efficient methods that could prevent the success of such an attack with absolute certainty. Berthod and Langos [22] proposed that intersection attacks can be mitigated through the use of dummy messages that create a constant flow of data throughout the network.

Timing attacks are more applicable to systems with very low latency than to higher latency systems such as that proposed in this work. Passive timing attacks, that view a conversation between two users, are difficult to execute because of the delays that can be inserted into high-latency systems. Active forms of timing attacks, specifically ones that involve traffic shaping, are also difficult to execute for the same reasons.

The Sybil attack can be combatted through administrative oversight in the user application process. In [11], social networking is described as a tool that might be used when determining user trust. By allowing connections to be formed based off of how well a user is known by other users one can prevent adversaries from creating multiple pseudonyms that link to a single user.

One portion of attack used in several of the above methods is traffic analysis. In [23], Venkitasubramaniam proposes a mixing strategy that makes it very difficult for an attacker to associate incoming messages with outgoing based off arrival time. In this strategy received messages are not sent in any particular order but are instead selected such that all messages are guaranteed to be sent at least by the time a number of messages equal to the buffer size have arrived.

2.2.4 Improving Anonymity Networks

Part of the purpose of reviewing current work is to use that research to improve upon older designs or to meet challenges proposed by a new system. In the case of applying Chaum Mixing to SMS there are several challenges. Two of the main concerns are authenticating a message given its size (as these are limited to 1220 bits in SMS) and power consumption (as mobile devices are limited in this regard). Fortunately, both problems can be tackled with a single solution, Elliptic Curve Cryptography (ECC).

In [24], Grillo et al. proposed applying ECC to SMS messages as a form of signature to authenticate a message. ECC is useful as a method of signing SMS messages because ECC permits shorter keys (a 160-bit ECC key is equivalent in strength to a 1024-bit RSA key), and is comparably faster in operations that involve private keys. This smaller key size allows greater flexibility when sending messages of different sizes because the required block size can be made to be smaller while also allowing a signature. ECC is also well suited as it has been shown to consume less power than other methods of encryption [25].

3. APPROACH

3.1 High Level Design

As mentioned earlier, the process of development could be most easily divided into three distinct logical categories, the first of which being theory. Theory involved selecting key aspects of the application including the base communication system on which to base the anonymous communication. In the end, this resulted in a mobile anonymity framework based on the Chaum Mix. At the highest level this network is composed of a small to medium sized group of smart phones about which no assumptions are made (other than their ability to use SMS and run user generated code). Through the use of user-generated code provided by this work, each of these phones can act as both a sender and as a mix in the network. These devices or actors each have the same capabilities and can communicate with each other in a peer-to-peer fashion. As communication makes use of ECC it is also necessary to have some way to distribute a list of phone numbers and associated public keys. As such, there also might exist some credential server that provides each actor with said list (though these could also be distributed in another fashion). This results in the network shown in Figure 3.

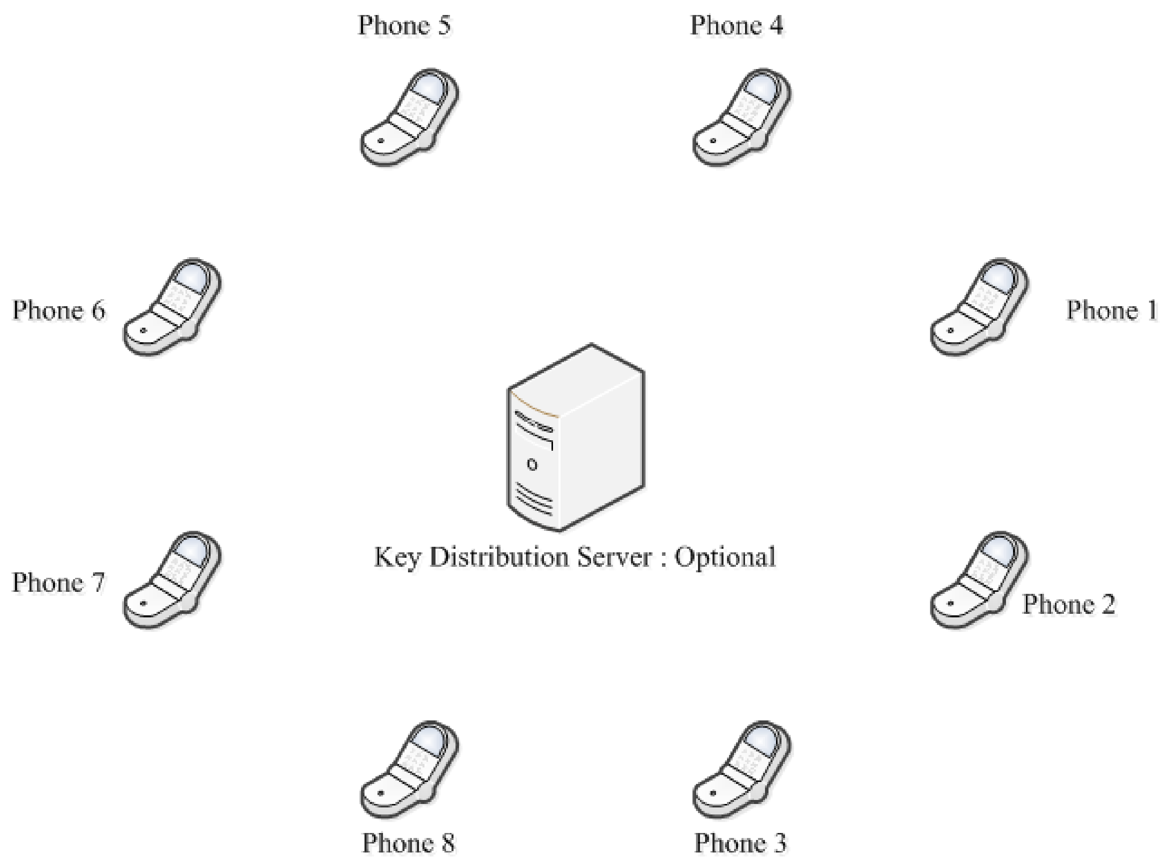


Figure 3: A top-level abstraction of the actors in the SMS Mix

3.2 Theory

In order to begin the design of the network it was decided that a good deal of time could be saved if said network could be based off of or made to extend a previously designed and tested network. The Chaum Mix was selected for this role as it designed for electronic mail and can be applied to any communication protocol that functions by sending distinct messages. The proposed system achieves anonymity by first identifying key characteristics of Chaum's algorithm and then applying those characteristics to a protocol appropriate to SMS and the types of messages that it generally carries. In this case each SMS device plays the role of both Mix and actor, an SMS message plays the role of a data packet or piece of electronic mail, and the underlying cellular network acts much like a traditional computer network. The protocol for communication is enforced by software running on each device. The system is described in its most secure form below, though several parameters (adjustable by group administrators) that improve efficiency but reduce security are mentioned.

Encryption is needed to obfuscate message paths as they pass through mix networks and hide message contents from potential observers. As in Chaum's work, encryption would be based on a set of keys. These keys would be available as a list with each key corresponding to the phone number of a single SMS capable mix device. While this list could be made available through some secure server the method of distribution is unimportant as long as it is secure and private. As SMS payloads are limited to 1120 bits, Elliptic Curve Cryptography is used for all encryptions as it achieves comparable strength to algorithms such as RSA while requiring much smaller keys (and thus smaller block sizes). Encrypting in layers, as is done in the Chaum network, is inappropriate for SMS communication as each layer of encryption requires some of the available data to be sacrificed for use in random bits. As an alternative, the SMS Mix uses a two-step approach. First, the message is encrypted for the final recipient. Once the message is encrypted and a route selected (as will be discussed later) the sender appends a signature, a route number, and some random bits. Finally, the sender encrypts this payload with the public key of the next mix, and sends the message through the network. In the case of a single mix, once the message is received, the mix will decrypt the message, remove the route number and random bits, attach a new tuple, and then send the message to the intended recipient. A diagram of a typical mix packet is shown below in Figure 4.

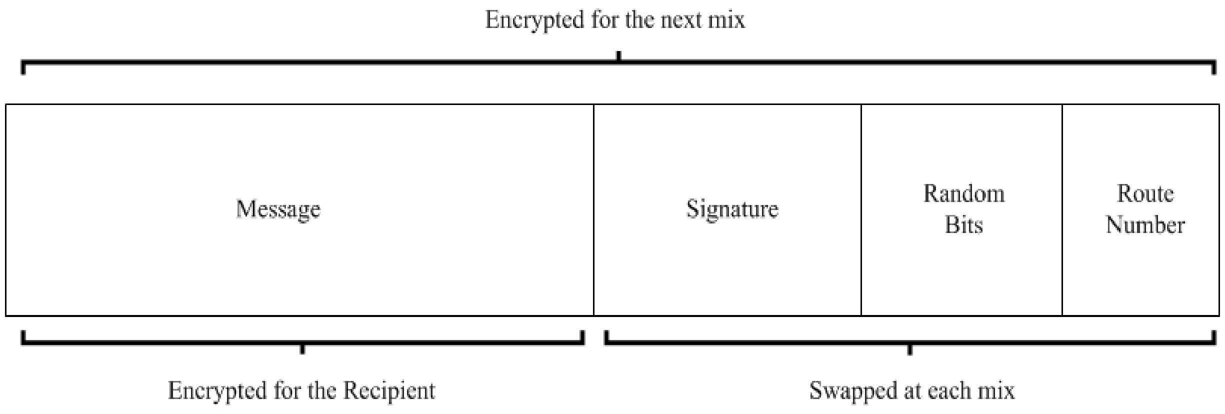


Figure 4: A typical SMS Mix packet

When a cascade of mixes is selected, the process is similar. The initial message is encrypted in the same way as described above, appended with the mentioned data, and then encrypted for the first mix. Once the message is received, the mix will decrypt the message, remove the route number and random bits, attach a new tuple, and then send the message to the next mix. As in Chaum [3], the mixing process repeats itself until the message reaches the end of the mix cascade, at which point the final mix will send the message to the intended recipient. In Figure 5 we see an example of encrypting in a cascade. One should note that the sending protocol for a single mix and a cascade of mixes are the same and that, regardless of the number of mixes in a cascade, the available bits in a message remain. This is because, prior to sending any messages, the sender negotiates a route with the system, reducing the amount of data necessary for encoding address information.

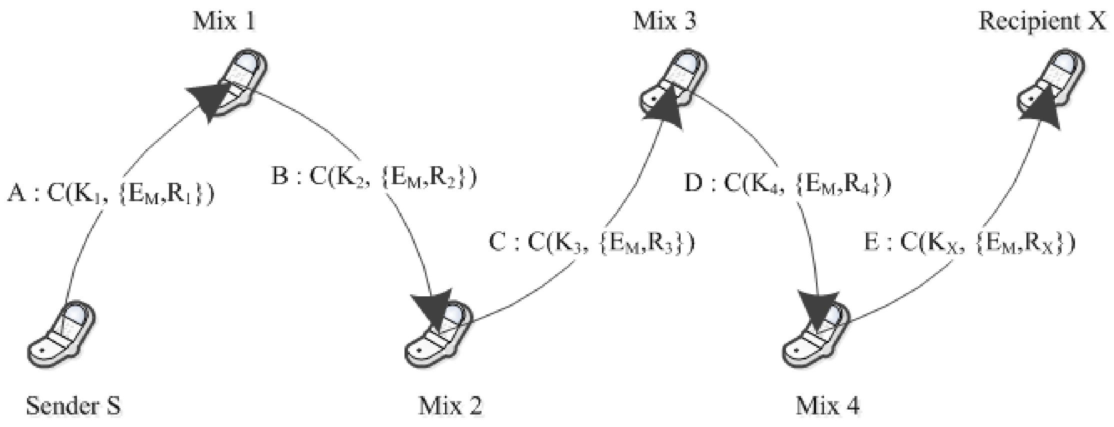


Figure 5: Sending an encrypted message

Route negotiation is similar to that used by other anonymity networks such as TOR [4] and is shown in Figure 6. When one user of the network wishes to send a message to another member, a number of mixes are selected randomly by the sender as well as an ordering for how messages should be sent.

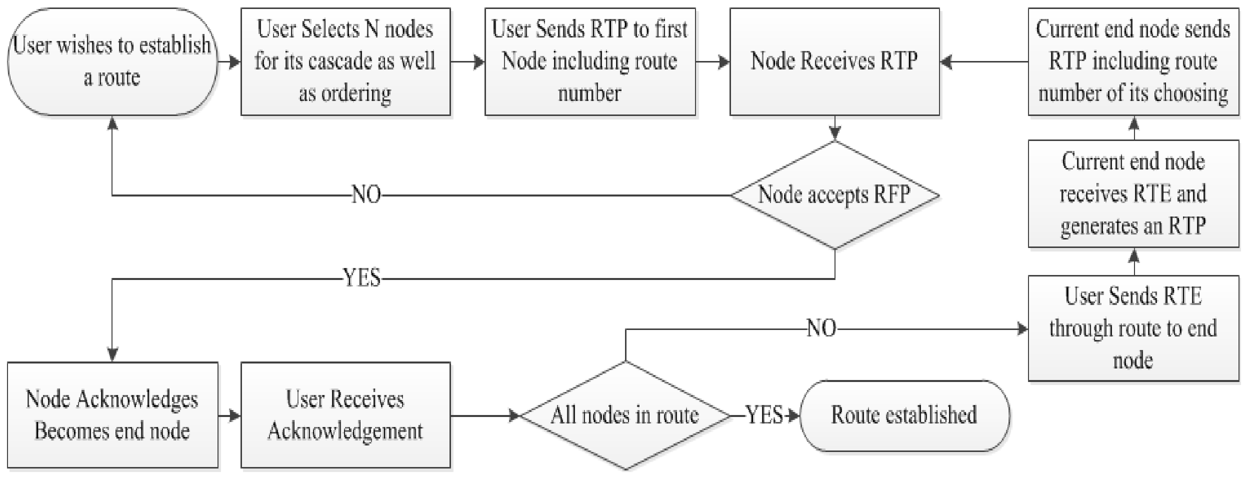

User wishes to establish a route
User Selects N nodes for its cascade as well as ordering
User Sends RTP to first Node including route number
Node Receives RTP
Current end node sends RTP including route number of its choosing
Node accepts RFP
NO
Current end node receives RTE and generates an RTP
YES
User Sends RTE through route to end node
NO
Node Acknowledges Becomes end node
User Receives Acknowledgement
All nodes in route
YES
Route established

Figure 6: Negotiating a route

Communication relating to route negotiation is handled in the same way as normal anonymous communication between two users but involves two three special kinds of messages: the RTP, the RTE, and the CRTP. The RFP, or request to participate, is a message that asks a particular Mix X to join as the current end of a route. The RTE, or request to extend, is a message that asks Mix X to extend a route by sending an RTP to a mix specified in the RTE. The CRTP, or confirmation of request to participate, is an acknowledgement message sent by a mix when it accepts an RTP.

In a typical case of route negotiation, a sender, labeled S, will create a RTP message asking a mix, labeled X, to participate. The RTP will be encrypted appropriately (including a tuple), and then sent to X. A portion of the RTP contents includes a route number. If X agrees to participate, it will associate the phone number the message came from with the route number it received and acknowledge that it has done so by sending a CRTP. This acknowledgement will include a digital signature that only X can create. From this point on, any message with that combination of phone number and route number will be treated as a member of that path or route. This identification is bidirectional; S and X will use this route number when the stream of messages is flowing from S to X and vice versa.

After receiving the first CRTP, S will send an RTE to X providing an address of a mix Y, to add to the end of the route. X will then select a route number to be used between X and Y, and then send an RTP to Y thus beginning the process again. When Y sends a CRTP, that message will be carried all the way back to S confirming the newest addition to the route. This process will repeat until the recipient, labeled R, has been reached. At this point, a RTP is sent to R but not an RTE. If any node is participating in a route but does not connect to any further nodes, it may assume that it is the intended recipient. From this point on, S can then send messages to R. During negotiation each mix stores a 4-tuple containing two phone numbers and two route numbers corresponding to messages coming from either connected Mix in a route. As such it is not necessary for S to attach each address as was originally proposed by Chaum. Instead, S needs only append a route number for the first mix to which it sends. Each subsequent mix will remove the route number that was received and add the appropriate number for the next recipient. This 4-tuple also allows R to respond to messages sent by S simply by sending a message through the route. Since negotiation uses the same set of messages for all stages, it is impossible for a mix to know anything about its position in the route other than its immediate predecessor and the Mix directly following it.

In this work, digital signatures are created using the Elliptic Curve Digital Signature Algorithm (ECSDA) as proposed by Grillo et al. [24] because ECC permits shorter keys (a 160-bit ECC key is equivalent in strength to a 1024-bit RSA key), and is comparably faster in operations that involve private keys. By using ECDSA based off of the proposed curve P-192 a user is left with a 48-byte key, leaving 92 bytes for a message data payload. ECC is also selected as it has been shown to consume less power than other methods of encryption [25]. In this work, each mix acknowledgement is signed with a digital signature that prevents a compromised mix from disregarding the address specified in the RTE. Messages sent back and forth between the users are also signed to guarantee the validity of the sender.

Unlike in Chaum's initial proposal, the SMS Mix will not send batch messages. As each message can only be sent one at a time, and the size limitations of such messages disallows data from multiple users to be included in a single message, an alternative to batch messaging was selected. This alternative was for the SMS Chaum Mixnet to send messages on a time quantum. In such a system, all received messages must wait until the end of the current time slot to be sent. When a slot does expire, a number of messages marked to be sent are transmitted in rapid succession, and in some random order. In the case where the number of messages waiting to be sent is below a certain threshold, the system will create dummy messages and forward them to random phone numbers within the group. These messages would be assigned a route number reserved for dummy messages. As all messages are encrypted, an outside observer would detect no difference between a dummy message and a real message. Devices in the network will discard any message with the dummy route to save buffer space. When each device maintains a steady rate of sending, be it real or dummy messages, it becomes difficult for an observer to gather statistical data based on sent and received messages. Network appearance to an outside observer and to a network user is shown in Figures 5 and 6 below.

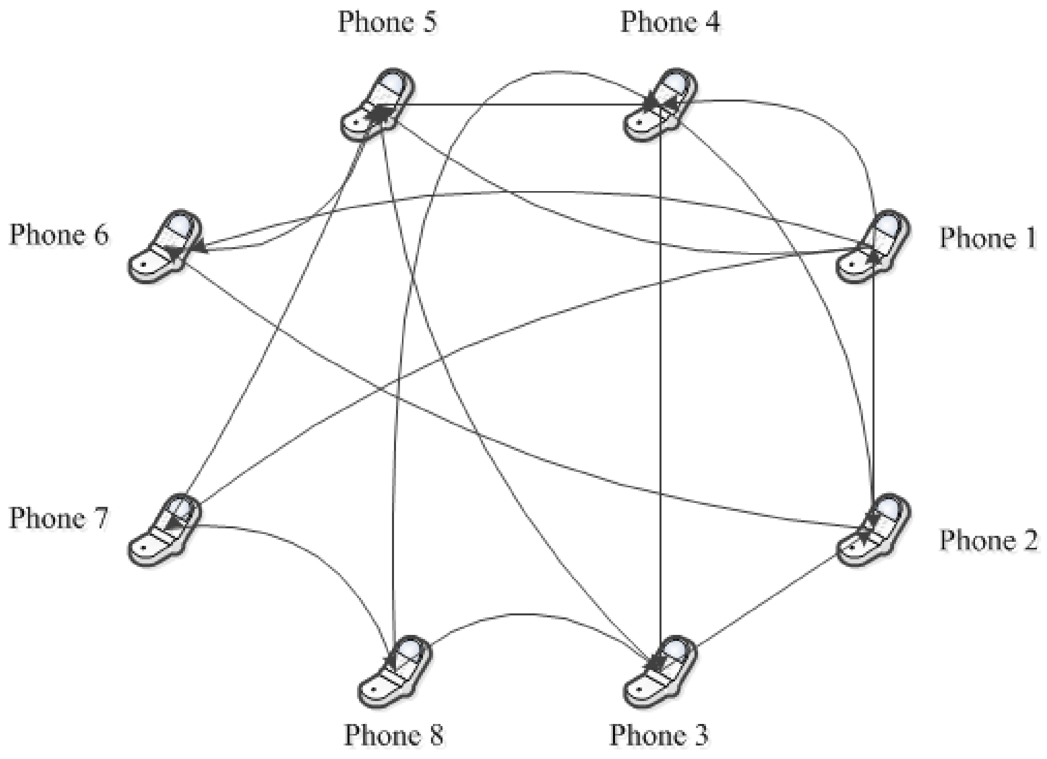


Figure 7: An SMS Mix from an outsider's perspective

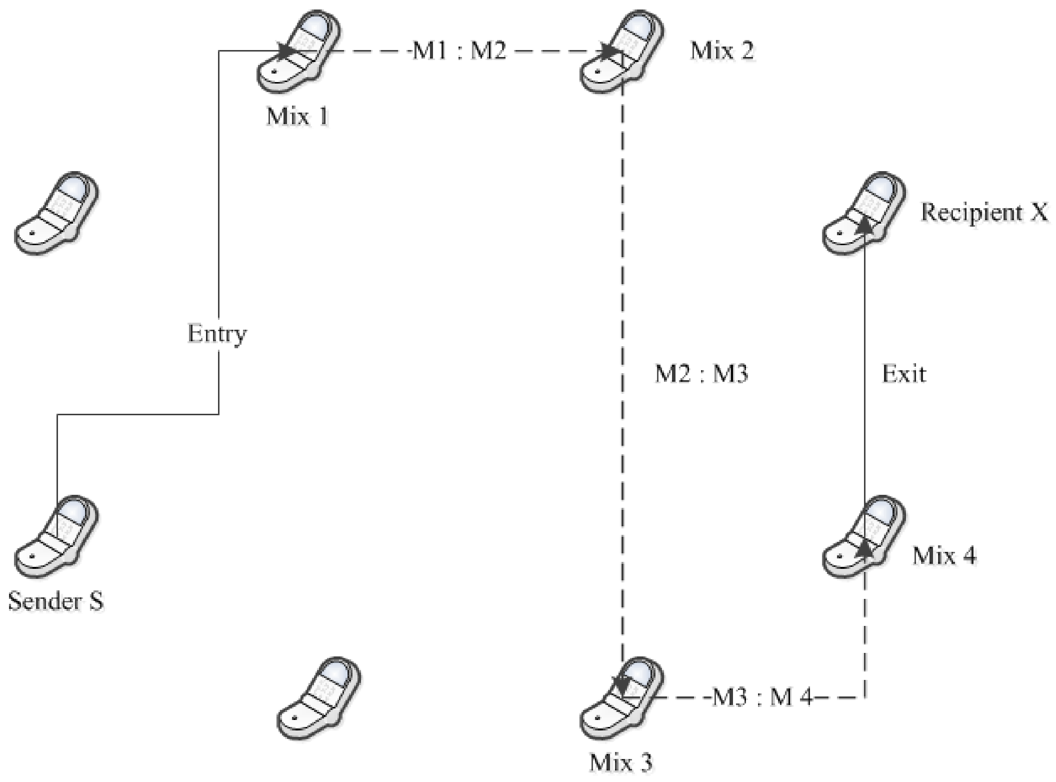


Figure 8: An SMS Mix from the user's perspective

As with the original Chaum Mix, nodes must also prevent duplicate messages or repeats from being processed. This can be accomplished by taking a hash of received messages (before decryption) and then comparing new messages to that hash, discarding any that match.

Research conducted following Chaum's initial work has noted several security weaknesses that appear in anonymity networks as well as solutions to them. After reviewing these weaknesses and solutions it is imperative that the Chaum SMS Mix address each to the fullest possible extent in order to prevent the system from being compromised. This work discusses solutions and how they would be applied to the SMS Chaum Mix below.

N-1 attacks as directed against the SMS Chaum Mix rely on two key assumptions. An attacker must be able to flood a system with messages and then, after a buffer flush, an attacker must be able to pick the message in question out of the remaining batch. In order to make flooding almost impossible, this work places two requirements on sending messages. First, in order for any messages to be sent and not immediately discarded they must be on a pre-negotiated route. Second, in order to be allowed to create a route in the first place a node must be in the privately available list of users and keys. Since each group uses a small list which is maintained by the group itself, an adversary must compromise the list or infiltrate the group to set up a route. If an adversary cannot accomplish either of those things, then a route cannot be established and a message flood cannot be initiated. In order to make message identification difficult in case a flood does occur, the proposed system generates dummy messages.

A dummy message is a message containing random text whose route number is set to a reserved unused route. Every batch sent out by a mix will contain some amount of dummy traffic with the possibility of a batch occasionally containing only dummy traffic. This traffic has two main benefits. First, in the case of a message flood, an attackers messages would not be the only ones in a batch. Second, these messages help maintain a constant flow of messages through the system. As the SMS Chaum Mix is designed for use by small groups, it is impossible to guarantee that the numbers of messages present in the system is significant in relation to the number of possible mixes. Without these numbers, a mix is forced to send out very small batches or hold a message for long periods of time, which increases succeptability to the n-1 attack. In the SMS Chaum Mix, each real message received is placed into a queue. When a group of messages are to be sent, the system will randomly select a number of slots in the batch for dummy messages addressed to random Mixes and then fill in the rest of the batch with real traffic. Real and dummy traffic is then sent from the mix into the system. Recipients of dummy messages will identify the reserved route number and immediately discard the message uppon decryption. Since dummy messages originate from within a mix, if a user does face an n-1 attack then an adversary, even one that could restrict incoming messages to a mix, would not be able to pick the target message out of a batch.

To combat the trickle attack the system makes use of the same methodology as above with a single addendum. If there are not enough messages to make a full batch then extra dummy messages are added until the batch is of the decided static size. In this way, even if an attacker only allows a single message through, the batch size of a mix remain unaffected. Furthermore, the system will randomly select some batches with a certain frequency to contain only dummy messages to further hide real traffic.

In order to combat the predecessor attack, the SMS Chaum Mix builds off of the ideas of Wright et al. [19,20,21] as mentioned in Chapter 2. System administrators would select a small number of devices for this role and then allow all users to begin their connections through these devices. In a use case scenario, administrators could purchase several cash SMS enabled phones, install the anonymity software and connect them to the network. These guards could be secured in different locations and then connected to a power supply. In order to keep these guards from being a single point of failure in the system, users will not be required to connect through them, but will try to do so first. Though the devices are designated entry guards, they will still be available for use as normal mixes in the network. If this was not the case the knowledge of which devices are entry guards could provide information about a devices location in a route.

Intersection attacks are very difficult to combat. Berthod and Langos [22] proposed that intersection attacks can be mitigated through the use of dummy messages that create a constant flow of data throughout the network. In order to combat this attack, SMS Mixes again make use of dummy traffic sent on a time quantum. If all users are communicating, then the intersection attack becomes more difficult as an attacker cannot use offline users to reduce the size of the searchable subset. The probability for any user to receive a number of dummy messages follows the binomial distribution:

$$F(k;n,p) = \binom{n}{k} p^k (1-p)^{(n-k)} \tag{9}$$

where n is the total number of dummy messages sent, k is the number of messages received and p is the probability of receiving a dummy message in a time quanta. This the probability can further be decomposed to:

$$F(k;n,p) = \binom{x*s}{k} \left(\frac{1}{x-1}\right)^k \left(1-\frac{1}{x-1}\right)^{(x*s)-k} \tag{10}$$

where x is the number of participants, s is the number of messages each user sends. The probability p becomes 1/(x-1) because each user selects the recipient of a dummy message from all other users. The probability of receiving at least 1 dummy message is then the probability of recieving no dummy messages subtracted from 1:

$$1-P(NoMsgs) = 1-\binom{x*s}{0} \left(\frac{1}{x-1}\right)^k \left(1-\frac{1}{x-1}\right)^{(x*s)-k} \tag{11}$$

Based off of the number of participants, the users could modify the number and ratio of dummy messages in a batch until the probability of any single user receiving a number of dummy messages was significantly high. The number of dummy messages sent would directly impact performance as either the number of messages sent in a batch would need to increase or the ratio of real to fake traffic would change.

Along with sending dummy messages, the proposed system also allows routes to expire after a given amount of time. When a route expires the user has to renegotiate a route to continue to send messages through the network. Since intersection attacks rely on the idea of gathering sets of messages that only intersect once, and using that to identify either a communication partner or the next mix towards that partner, renegotiating routes would force an attacker to begin an unfinished intersection attack again.

Timing attacks are more applicable to systems with very low latency than to SMS Chaum Mixing. Because of the variable delays introduced by users and the network, and because all messages are sent on time quantums, accurate timing information becomes very hard to identify. As both passive and active timing attacks require this information to be effective, the underlying properties of the system act as an adequate defence. As such the SMS Mix does not need to directly address these kinds of attacks in its design.

The Sybil attack, which requires a single user to create multiple identities, is difficult in SMS Chaum Mixing because the proposed system is designed for use by small groups. These groups often who can personally add devices to the network and have some oversight into this process. This oversight prevents a user from adding multiple identities to the network. As mentioned in [11], it is quite likely that users in the network have social connections to each other and would be able to police their network for intruders with little difficulty.

Another form of attack that needs to be addressed is traffic analysis. Similar to the timing attack, traffic analysis involves an adversary passively observing a network of mixes. An adversary will observe the inputs and outputs of a mix in an attempt to create source-destination pairs for these messages. In [23], Venkitasubramaniam proposes a mixing strategy that demonstrates asymptotic optimality. In this strategy a number of packets equalling 1/3 of the buffer size are set aside. The remaining packets are divided into two groups (1 and 2) also equal to 1/3 of the buffer size. The distribution of packets from a mix-route pair in each of those groups, assuming unequal arrival rates, is proportional to the multinomial distribution:

$$Pr\{m_1,\ldots,m_k\} \propto \begin{pmatrix} \frac{B}{3} \\ m_1 \ldots m_k \end{pmatrix} \lambda_1^{m_1} \ldots \lambda_k^{m_k} \tag{12}$$

where (m1 ... mk) is the set of possible compositions, B is the total buffer size, and   is the arrival rate from a mix-route pair i. Once the buffer is filled, the first third of the packets that were set aside as well as group 1 are combined and marked for sending. As each new message arrives one of the messages from the marked group are sent based on some random ordering. After all messages marked for sending have been sent, the messages that arrived are again divided into two groups and the process repeats. The proposed system uses this strategy for sending messages, allowing the user to set the buffer size to some constant value across the system. A summary of the mentioned attacks and their applied solutions is shown in Table 1.

Table 1: A summary of attacks and solutions for the SMS Mix

Attack Name	Attack Summary	Attack Description
N-1 Attack & Trickle Attack	The attacker forwards fake traffic with a single legitimate message or the attacker delays all but a single legitimate message from entering a mix.	• Limit number of users to a small group whose participants approve members • Require membership in order to send messages • Use dummy traffic ratios to prevent one user's messages from dominating a batch
Predecessor	Given enough path reformations, an adversary eventually becomes the predecessor to the sending node, allowing an attacker to gather data used to identify a sender.	• Select a small number of trusted nodes as entry guards. All communication routes begin through these nodes
Intersection	An adversary takes the intersection of all recipients of all Batches suspected to contain a message from a selected user. Eventually only one recipient remains.	• All users communicate via real or dummy messages at each time quanta, thus preventing an attacker from determining which users are using the network at which times • Reform routes at set intervals
Timing	An attacker shapes traffic to form observable timing metrics through the system.	• High traffic latency • Fixed sending rate
Sybil	A single user controls multiple nodes in the system, giving more influence then should be allowed and allowing for information collusion between members of a route.	• Limit number of users to a small group whose participants decide who can be in the group • Require membership in order to send messages
Traffic Analysis	An attacker observes all messages entering and leaving a node and uses this information to correlate senders and receivers	• Dummy Traffic • Venkitasubramaniam's asymptotically optimal mixing strategy

The above system is designed specifically to be as secure as possible. However, it is recognized that the addition of security comes at the cost of throughput. In order to make the system flexible several security paramaters can be adjusted that reduce security but inprove overall operation. Group administratiors would be able to do the following: enable or disable the sending of dummy messages, adjust the ratio of real to dummy traffic, adjust the minimum batch size, set a maximum and minimum route length, and finally limit the traffic a single user can generate.

3.3 Code Development

Once theory detailing how the SMS Chaum Mix would operate was made concrete, the development of code began. Before the program was placed on any mobile device, it was decided that it would be wise to write and test the algorithm in a more controlled environment. To this end, the algorithm was written in java code and run within a specially designed simulator. This simulator played the role of the cellular network as well as acting as the mobile device hardware. Within the simulator were instances of virtual mobile devices. Each of these instances contained a reference to the algorithms code and could execute it in response to messages delivered by the simulator. These devices used the same interface as SMS in order to facilitate code migration later. When the full simulation was complete, it allowed for full functional testing of each of the algorithms in the theory section without having to rely on the cellular network or risk incurring unforeseen cellular expenses had there been some error in the code. This also allowed for rigorous unit testing of the mix network.

Once the algorithm had been satisfactorily tested, the code was moved to the Android mobile operating system. This involved migrating the protocol code as well as developing the application front end to handle user input and a service back end to interface with the device radio for the purpose of sending and receiving of messages. This was fortunately quite simple because Android uses a version of the Java programming language very similar to that which runs on personal computers.

Android makes user of the open source BouncyCastle library to handle encryption. Unfortunately, Android only includes a portion of that library in its operating system code. Specifically absent is the code for Elliptic Curve Cryptography. As Android includes BouncyCastle in its build path, including the full BouncyCastle causes naming conflicts. In order to make up for this an external library called SpongeyCastle was used. SpongeyCastle is a repackaging of the BouncyCastle library edited slightly to run on Android, and provides all of the functionality of BouncyCastle. SpongeyCastle, like BouncyCastle, is open source.

In order to make sure that the application could run on a wide range of phones, the application was built with backwards compatibility excluding only the very earliest versions of the operating system. This was important as many low cost phones receive few if any operating system updates from a carrier. Given the lifetime of a smartphone, it becomes very likely that many of the devices currently in operation have not been updated to the most recent version.

3.4 Testing

The final step in the approach taken by this work was application testing. This testing was done in addition to the initial testing of the protocol on personal computers and was directed at confirming the success of the code migration from the Java virtual machine running on most computers to the version of Java running on the Android operating system. As there are many different models of phone which run the Android operating system, it was necessary to carry out testing across multiple types of smartphones running several versions of the operating system to verify compatibility with both the software and the hardware aspects of the phone. Testing also involved taking data for power usage while the network was and was not sending messages. This data will be presented in Chapter 4.

4. RESULTS AND ANALYSIS

The results of this work can be split up into two parts, theoretical and practical. The theoretical aspects of this work, in a broad sense, involved designing and planning a system that made use of the ideas presented in Chaum Mixing as well as improvements developed since that point. This process began with identifying key aspects of message mixing that were common to all frameworks based on this principle. These aspects were then applied to the SMS capabilities present on any phone. Once this network was drawn out it was important to take into account known attacks as progress is made by building upon the work of others instead of reinventing it. This involved cataloging and classifying attacks as well as developing theoretical solutions that could be implemented on the designed system. Lastly, one had to alter the way a normal mix works in order to take into account the limitations of SMS, all without compromising its security. The result of this process was a well-defined theoretical architecture for an SMS based Chaum Mix.

The practical aspect of this work involved building and designing the described system that could be installed onto smartphones. This involved learning how to program for Android phones as well as understanding the intricacies of the Java programming language. The result of this work was a functioning system that allowed for mixing among participating smartphones.

4.1 Challenges

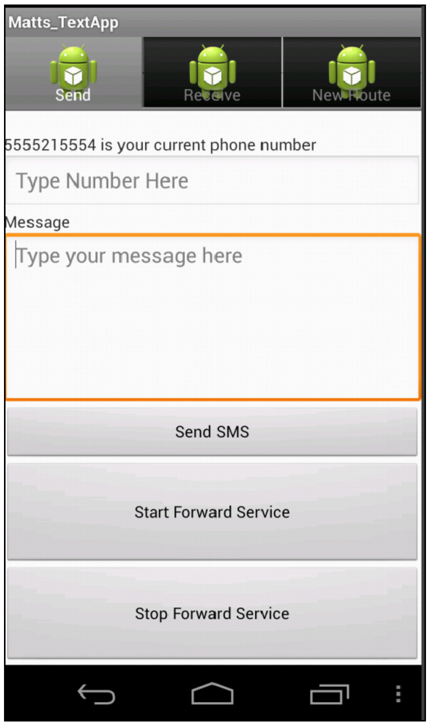
There were many challenges associated with the completion of this work. A few of the most difficult ones are discussed here. In the theory portion of the work, the greatest challenge was gaining a clear and concise understanding of decades of work in the fields of privacy, anonymity, and secrecy. This involved grasping complex algorithms and advanced applications of probability and statistics used by attackers to break the anonymity these algorithms were attempting to provide. As many of these algorithms relied on cryptography it was also important to understand many of the related mathematics and concepts.

Several challenges also appeared in the practical design phase of the project. Initially, selecting a cryptographic algorithm proved difficult. This was because most algorithms, in order to provide any marked level of security, require keys and block sizes that are larger than what can be transmitted by a single SMS message (1120 bits). In addition to size requirements, it is necessary to minimize the power consumption of the cryptographic algorithm executed in memory (since few if any Android phones include dedicated cryptographic circuits). These needs led to the eventual selection of the already existing Elliptic Curve Cryptography by the author of this work. After selecting ECC, it was important to understand how it operates. This was quite challenging in and of itself as the theory behind ECC is quite complicated, requiring an understanding of a branch of mathematics usually only found in very niche domains.

Aside from understanding the theory behind ECC, its implementation also proved difficult. This is because the cryptographic library used by Android, known as Bouncy Castle, does contain ECC functionality. More specifically, Bouncy Castle does contain ECC functionality, but that functionality has not been included in the most recent builds of the operating system. The fact that Bouncy Castle was partially present in Android presented further problems as it actually prevented a user from including the full Bouncy Castle library. This was because including the library caused naming conflicts in the Java class loader (there were two libraries named BouncyCastle) that prevented compilation. Fortunately the library had been repackaged under another name (SpongyCastle), which allowed inclusion of the missing functionality. Even after the inclusion of this library ECC proved challenging as there was few examples detailing proper usage of library APIs.

4.2 Resulting Application and Screens

After varying degrees of effort, each aforementioned challenge was eventually overcome and design of the system was completed. The Android application consisted of several screens that allowed users to establish routes and send and receive messages. The design of the application screens did not incorporate any custom UI and was meant to simply act as an interface to the underlying functionality instead of focusing on elegance or, to some degree, user friendly interaction. Images of the device screens are displayed below in Figures 10 through 17.

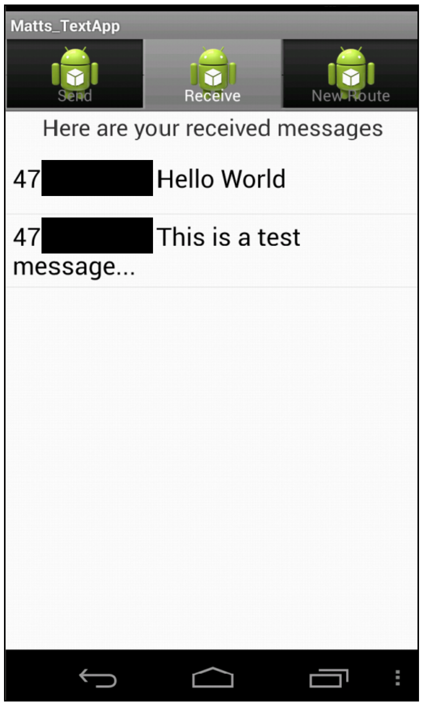


5555215554 is your current phone number

Type Number Here

Type your message here

Figure 13: The SMS Sending Screen

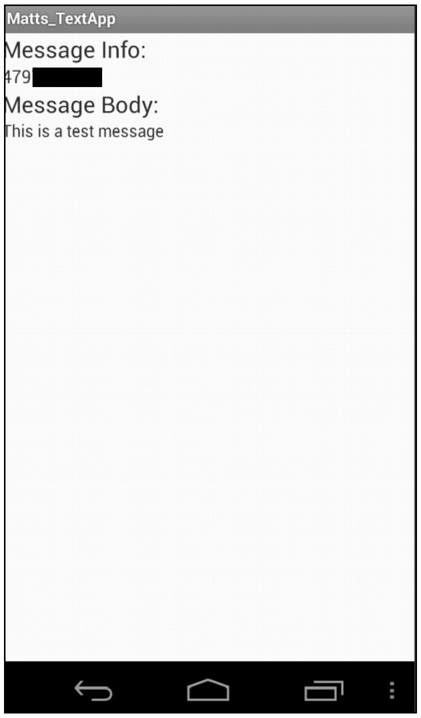


Here are your received messages

47[redacted] Hello World

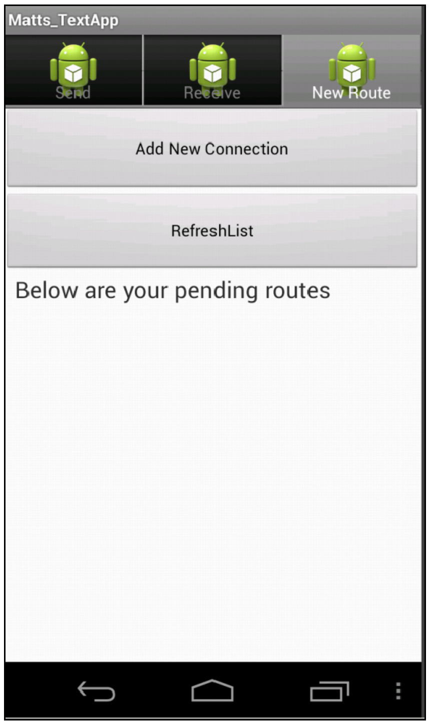
47[redacted] This is a test message...

Figure 14: The Received Messages Screen



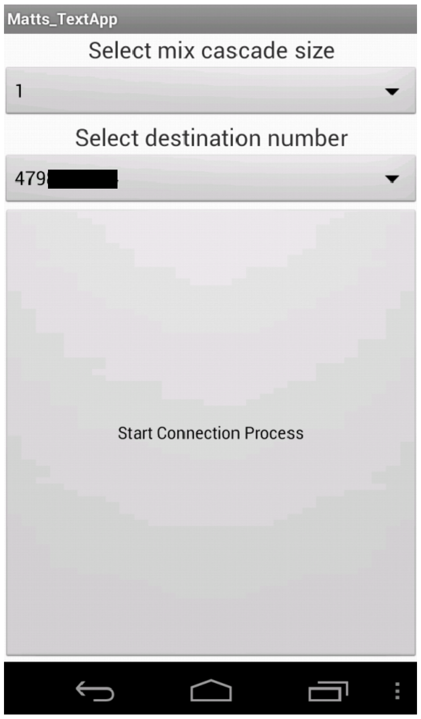
Message Info:
479[redacted]
Message Body:
This is a test message

Figure 15: The Individual Message Screen



Below are your pending routes

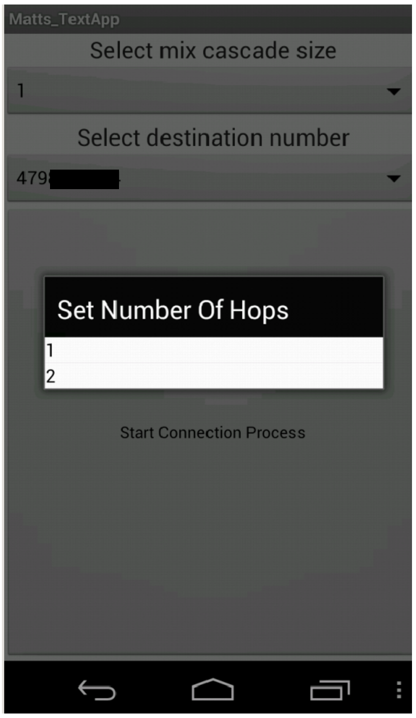
Figure 16: The New Route Screen



Select mix cascade size: 1

Select destination number: 479[redacted]

Figure 17: The Route Generation Parameters Screen


Matts_TextApp

Select mix cascade size

1

Select destination number

479[redacted]

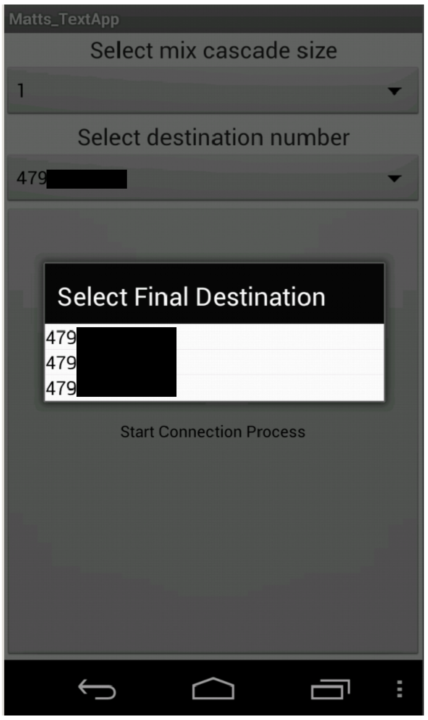
Set Number Of Hops

1

2

Start Connection Process

Figure 18: The Route Generation Parameters Screen (Cascade Size)


Matts_TextApp

Select mix cascade size

1

Select destination number

479[redacted]

Select Final Destination

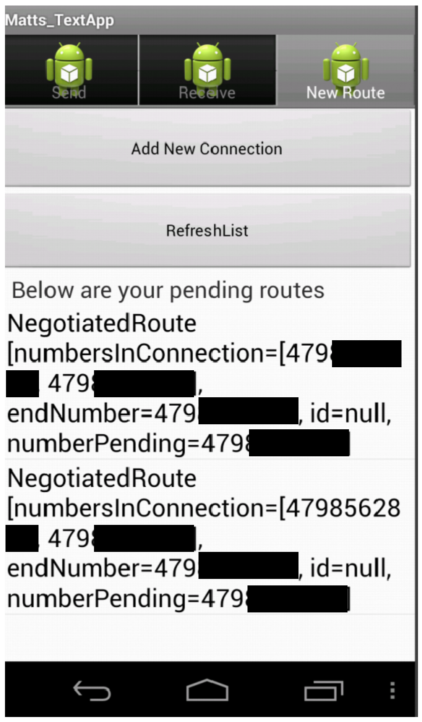
479[redacted]

479

479

Start Connection Process

Figure 19: The Route Generation Parameters Screen (End Number)



Below are your pending routes

```
NegotiatedRoute
[numbersInConnection=[479[redacted]
[redacted] 479[redacted],
endNumber=479[redacted], id=null,
numberPending=479[redacted]
NegotiatedRoute
[numbersInConnection=[47985628
[redacted] 479[redacted],
endNumber=479[redacted], id=null,
numberPending=479[redacted]
```

Figure 20: The Route Generation Parameters Screen with Pending Routes

4.3 Testing Methodology

In order to verify system functionality a methodology for testing was developed. As mentioned previously, this involved installing the application on several different phones, establishing routes, and then sending messages across those established channels. Attempts were also made to send and receive messages that did not follow the system's established protocol. This was made up two broad cases: messages from unverified sources, and messages that did not follow protocol.

The test configuration was composed of 4 phones across three hardware models and three versions of the android operating system. Though there were 4 phones, only three phones were present in the network at any given time (as this was the maximum number of phones licensed for the cellular network at any given time). These phones were used as follows: two Samsung

Captivate cellphones running android 2.1 (froyo or frozen yogurt), one Motorola Atrix 4g running android 2.3 (Gingerbread), and one Samsung Galaxy S4 running android 4.3 (JellyBean).

Using these hardware and software configurations several network tests were run to verify functionality. The methodology for a single run of these tests was as follows:

1. Each phone was turned on and given a list of phones present on the network
    - a. This list also included public keys of said phones
2. One phone was selected to be the initiator and the other to be the final recipient
3. The initiate route button was pressed on the initiating phone
4. Messages relating to route generation were viewed and verified
5. On completion of route generation, messages were sent from connected phones
    - a. These messages were monitored to ensure correct content
    - b. Dummy messages sent by the phones were also monitored
6. After messaging was confirmed to be successful, illegal messages were sent
    - a. These messages were sent by the native messaging applications
    - b. These messages were illegal because they
        - a. Did not conform to the protocol specification
        - b. Did not belong to any established route
        - c. Did not originate within the network list
        - d. Did not use the correct encryption scheme
        - e. Did not contain a correct ECDSA digital signatures
    - c. These messages were verified as being discarded by the network

4.4 Results and Analysis

When tested under normal conditions the system behaved within expected parameters regardless of hardware of software configuration differences. Routes could be established with consistency and messages sent arrived at their intended recipient following said routes. Dummy messages were, as was specirfied in the theory portion of the work, sent out by mixes and were immediately discarded upon being received. Additional delay in messaging speeds was experienced. Howerver, this was due in part to encryption and decryption, along wit h the fact that the message was sent by several phones instead of just one.

The system also performed as expected when tested under abnormal circumstances, specifically when messages were introduced that would be considered illegal by the system. Regardless of whether a message did not originate from a route, did not originate from a phone in the network, did not follow the correct protocol, or did not contain contents that were properly encrypted or signed, were the same. Specifically the system discarded incorrect messages.

5. CONCLUSIONS

5.1 Summary

In this work, a Chaum mix inspired, smartphone based network that uses the Short Message Service (SMS) is designed and developed. Its purpose is to provide anonymity for small groups of individuals numbering 30 to 40 members that wish to exchange short messages. The group maintains a small private list of phone numbers, and public keys. Each phone number corresponds to a smartphone, which by running code developed in this work, acts like a mix as described by Chum. By applying encryption and providing message routing each of these mixes allow users to maintain anonymity and keep the contents of their communication secret. ECC public key cryptography is used so that encrypted messages fit into the SMS payload and can be signed to provide authentication.

The code in this work was written in Java for the Android operating system and makes use of libraries present in that language. The code was originally written and tested on a personal computer and then migrated to the mobile phone platform after the functionality of the mixing algorithm was verified. The code was written to be backwards compatible to ensure that a maximum number of devices could run the software detailed in the work.

Testing on the phone was done across several hardware and software configurations. Other than testing to verify base functionality, a number of tests were also run in which incorrect inputs were given to the system to gauge how it would handle these cases. In each scenario the testing was successful.

5.2 Potential Impact

The proposed system provides a method of obtaining anonymous and secure communication to a previously unprotected medium, the SMS message. Though some technical knowledge is required to set up and maintain this system, day to day use is simple and requires no advanced knowledge. This is important as many users of electronic communication are not technically knowledgeable and have little to no understanding of privacy, security, and the vulnerability of their information. A system catering to that group not only fills that security hole, but if successful in the marketplace, also promotes other developers to adopt similar methods of design, resulting in a multitude of varied and easily used security options. These options deter monetarily focused adversaries as they make profit based off of information exploitation difficult. They also deter large entities such as governments from abusing their ability to observe communications unimpeded.

5.3 Future Work

It is understood that battery life is an important resource in the mobile domain and as such future work would involve empirical testing in order to quantify and mitigate power consumption caused by processing messages. In implementation, mitigation could be accomplished through algorithms or rule sets designed to reduce power usage. This could involve message throttling either through imposing sending limitations, limiting the number of routes a phone could partake in, optimizing dummy traffic, or attempting to assign high traffic routes to unused nodes upon route reformation (in order to load balance battery usage). Power consumption relating to encryption and its use could also be measured and mitigating techniques could be proposed.

Other than dealing with the physical constraints, future work could also focus on how the users interact with the application and how system administrators would go about setting up a network of users. In regards to interaction, the focus of the user interface thus far has only been directed at providing a way to interact with the underlying framework. Future work would consider and implement user friendly and attractive interfaces in order to promote wider use as a system that lacks those two qualities risks failing to interest its target audience, those with little or no technical knowledge. In regards to system setup, currently only the phone application portion of this system has been implemented, meaning that there is no implemented way to add users to the network or to easily share and distribute keys. Future work would focus on setting up a simple and secure way to add users, update user lists on mobile devices, and share keys. This would possibly take the form of a standalone application that might run on a server.

REFERENCES

[1] Pfitzmann, A. and Hansen, M. (2008) Anonymity, unobservability, and pseudonymity: A consolidated proposal for terminology, Version v0, 31, 15.

[2] Shannon, Claude E. (1949) Communication Theory of Secrecy Systems. Bell system technical journal, 28, 656-715.

[3] Chaum, D.L. (1981) Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms, Communications of the ACM, 24, 84-90.

[4] Dingledine, R., Mathewson, N. and Syverson, P. (2004) Tor: The Second-Generation Onion Router, Naval Research Lab Washington DC.

[5] Jerichow, A., Muller, J., Pfitzmann, A., Pfitzmann, B. and Waidner, M. (1998) Real-Time Mixes: A Bandwidth-Efficient Anonymity Protocol, IEEE Journal on Selected Areas in Communications, 16, 495-509.

[6] Goldschlag, D.M., Reed, M.G. and Syverson, P.F. (1996) Hiding Routing Information, Information Hiding, 137-150.

[7] Clarke, I. and Sandberg, O., Wiley, B. and Hong, T.W. (2001) Freenet: A Distributed Anonymous Information Storage and Retrieval System, Designing Privacy Enhancing Technologies, 46-66.

[8] Reiter, M. and Rubin, A. (1998) Crowds: Anonymity for Web transactions, ACM Transactions on Information and System Security, 1, 66–92.

[9] Freedman, M.J. and Morris, R. (2002) Tarzan: A peer-to-peer anonymizing network layer, Proceedings of the 9th ACM Conference on Computer and Communications Security, 193 - 206.

[10] Edman, M. and Yener, B. (2009) On Anonymity in an Electronic Society: A Survey of Anonymous Communication Systems. ACM Computing Surveys, 42, 5.

[11] Marques, R. and Zuquete, A. (2011) Social Networking for Anonymous Communication Systems: A Survey, 2011 International Conference on Computational Aspects of Social Networks, 249-254.

[12] Diaz, C. and Preneel, B. (2004) Taxonomy of Mixes and Dummy Traffic, Information Security Management, Education and Privacy, 217-232.

[13] O'Connor, L. (2005) On Blending Attacks for Mixes with Memory, Information Hiding, 39-52.

[14] Serjantov, A., Dingledine, R. and Syverson, P. (2003) From a Trickle to a Flood: Active Attacks on Several Mix Types. Information Hiding, 36-52.

[15] Reiter, M.K. and Rubin, A.D. (1998) Crowds: Anonymity for Web Transactions. ACM Transactions on Information and System Security, 1, 66-92.

[16] Wright, M.K., Adler, M., Levine, B.N. and Shields, C. (2004) The Predecessor Attack: An Analysis of a Threat to Anonymous Communications Systems, ACM Transactions on Information and System Security, 7, 489-522.

[17] Zhu, Y., Fu, X.W., Graham, B., Bettati, R. and Zhao, W. (2005) On Flow Correlation Attacks and Countermeasures in Mix Networks. Privacy Enhancing Technologies, 207-225.

[18] Douceur, J.R. (2002) The Sybil Attack. Peer-to-Peer Systems, 251-260.

[19] Overlier, L. and Syverson, P. (2006) Locating Hidden Servers, IEEE Symposium on Security and Privacy, 15.

[20] Syverson, P., Tsudik, G., Reed, M. and Landwehr, C. (2001) Towards an Analysis of Onion Routing Security, Designing Privacy Enhancing Technologies, 96-114.

[21] Wright, M., Adler, M., Levine, B.N. and Shields, C. (2003) Defending Anonymous Communications against Passive Logging Attacks, Proceedings of the 2003 Symposium on Security and Privacy, 28-41.

[22] Berthold, O. and Langos, H. (2003) Dummy Traffic against Long Term Intersection Attacks, Privacy Enhancing Technologies, 110-128.

[23] Venkitasubramaniam, P. (2010) Anonymous Networking under Memory Constraints, 2010 IEEE International Conference on Communications (ICC), 1-5.

[24] Grillo, A., Lentini, A., Me, G. and Italiano, G.F. (2008) Transaction Oriented Text Messaging with Trusted-SMS, Annual IEEE Conference on Computer Security Applications, 485-494.

[25] Potlapally, N.R., Ravi, S., Raghunathan, A. and Jha, N.K. (2006), A Study of the Energy Consumption Characteristics of Cryptographic Algorithms and Security Protocols, IEEE Transactions on Mobile Computing, 5, 128-143.