Multi-party and Distributed Private Messaging

by

Pratheek Nagaraj

S.B., Massachusetts Institute of Technology (2016)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2016

Author . . . .
Department of Electrical Engineering and Computer Science
May 19, 2016

Certified by. . . . .
Matei Zaharia
Assistant Professor
Thesis Supervisor

Accepted by . . . .
Christopher J. Terman
Chairman, Masters of Engineering Thesis Committee

Multi-party and Distributed Private Messaging

by

Pratheek Nagaraj

Submitted to the Department of Electrical Engineering and Computer Science
on May 19, 2016, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

In this thesis, I extend private messaging systems by designing multi-party private messaging models and contribute to the implementation of a distributed private messaging system. Private communication is of increasing interest yet current state-of-the-art adopted solutions are inadequate in providing both scale and privacy. Most current communication methods leak metadata or are susceptible to traffic analysis in spite of end-to-end encryption. Vuvuzela is a foundational private messaging system helps reconcile these two concerns. This project builds on Vuvuzela by introducing three group messaging models. These models tradeoff support for multi-party messaging with network bandwidth and latency. Additionally, this work describes the implementation and evaluation of key aspects of a distributed private messaging system, Stadium. This new system scales to more users while keeping server costs down to promote the adoption of private messaging as a more feasible practice.

Keywords. Privacy, Deniability, Messaging, Multi-party, Distributed Systems

Thesis Supervisor: Matei Zaharia
Title: Assistant Professor

Acknowledgments

I would first like to acknowledge my advisor, Prof. Matei Zaharia, for his invaluable guidance throughout this project and his supervision. Prof. Zaharia provided key feedback and suggestions into further areas of investigation in relation to this project. His insight in computer systems security and distributed systems has led my deeper understanding and appreciation of the discipline.

I would also like to thank Prof. Nickolai Zeldovich for his input in this work and with his assistance in familiarizing myself to previous work in the area.

This work would not have been possible without the assistance of Nirvan Tyagi, David Lazar, Yossi Gilad, and Justin Martinez. I worked closely with each of these individuals in understanding the existing private messaging systems as well as building a new system. This work would not have been possible without their collaboration.

I also acknowledge the CSAIL community and in particular the PDOS group whose support and network promoted the progress of this project.

Furthermore, I would like to extend thanks to the Department of Electrical Engineering and Computer Science at MIT. From the instructors who have been fundamental in my learning to the administration in the department who facilitated the my growth, the EECS Department is to thank for being my academic home over the past several years.

Finally, I am thankful to my family and relatives for their immense love and support throughout my educational experience thus far. In addition, I am grateful for my friends here at MIT and beyond who encouraged me and made the journey enjoyable.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Private communication over the Internet has become of increasing interest over the years. Users such as freelance reporters or whistleblowers require privacy in their communication and current systems to not provide an adequate guarantee. While information can be encrypted to prevent the uncovering of message data, *metadata* is still attached to such communications. *metadata* can include which users are communicating with one another and at what times they may be communicating, and metadata can be discerned by header inspection or traffic analysis. Recently, officials at the NSA have even stated that “if you have enough metadata you don’t really need content” [39] and “we kill people based on metadata” [25]. Thus protecting metadata in communications is a critical step to ensuring that communication is private.

Previous private messaging systems lacked the ability to handle a large number of clients or were computationally expensive. Vuvuzela, a scalable private messaging system, reconciles these two issues and this project builds on its foundational work.

1.1 Motivation

To motivate the need for private messaging consider a whistleblower, Alice, who wishes to expose corruption within her corporation or organization. If she wishes to go public with this information to a journalist, Bob, she would want her communications to be secure. If it can be determined that Alice is communicating Bob, then it raises suspicion that Alice may have leaked the information to Bob, even if the contents of her communications are never revealed. In fact, over a series of messages a stronger and stronger case can be constructed to incriminate Alice by analyzing the traffic patterns between Alice and Bob.

There is an ethical concern on both sides of private messaging. From the perspective of law-enforcement, private messaging is a bane to its ability to perform its duties. In particular, collective security is undermined when privacy, a personal good, is used for malicious activity. In this regard, surveillance of communication is essential to preventing the "bad guys from winning" [38]. On the other hand, from the perspective of privacy advocates and surveillance studies, private messaging is critical to personal freedom. Surveillance is seen as an instrument of power and privacy is to be seen as a social good. Mass surveillance leads to uniformity and a subdued population and alters individual behavior leading to fear and lack of freedom. Consequences of creeping surveillance include corporate and government interests as well as vulnerability to hackers [38].

This work does not take a stance on the ethical framing of private messaging, but rather addresses from a technical angle in order to raise the awareness of the discourse. The author's intent is for fair and good use of the technical work presented here and hopes that this work will contribute to the dialogue in a positive way.

1.2 Contributions

This thesis presents two main contributions:

1. The design and analysis of a multi-party or group private messaging system. This contribution examines extensions to the Vuvuzela system that would allow for communication to exist in more than just a bidirectional channel. This analysis paves the way for greater utility in private messaging systems as multi-party communication is often required for communication to be practically useful. The main result shows that while group messaging can exist, there is a notable performance penalty as a result of trying to conceal the multiple parties in the

conversation. Future work looks to investigate whether performance can be improved by alternative designs or by analyzing other tradeoffs that could be taken.

2. The implementation and analysis of a distributed private messaging system, Stadium [45]. The author, in conjunction with other Stadium contributors, helped developed a system that would allow a Vuvuzela-style private messaging system to scale to more users and reduce the network burden that a single server had to undertake to allow the private messaging system to operate. The author's contribution was largely in the implementation and testing of such a system. Future work aims to achieve better performance of Stadium by better parallelizing its operations in attempt to reduce the computational complexity a single server has to undertake.

Chapter 2

Background and Overview

Private messaging is not an entirely new line of research. Current private messaging systems exist but are largely unable to protect metadata for a large number of users.

One class of private messaging systems provide strong privacy guarantees. Dissent [47] and Ripose [11] are communication and computationally expensive requiring broadcasts of messages or intensive cryptographic operations such as Private Information Retrieval (PIR) [40]. These systems have only supported 5,000 users [47] or on the order of hundred messages per second [11] making them impractical for widespread use.

Another class of systems are more scalable. Tor [15] and other mixnet designs [8] suffer from traffic analysis or timing attacks. Furthermore, these systems require large numbers of users to support privacy guarantees. In addition, adding cover traffic to attempt to mitigate some of the security flaws in these systems make them more expensive and the tradeoff for privacy is still not fully achieved as adversaries can monitor traffic over time and active disrupt the network [2, 21, 13, 29].

Vuvuzela [46], is a scalable private messaging system resistant to traffic analysis that attempts to reconcile the current drawbacks of scalability and privacy. Vuvuzela provides a bidirectionally messaging framework allowing two parties to communicate. It provides differential privacy so long as there exists one honest server in the system. Furthermore, Vuvuzela can scale to millions of users and support tens of thousands of messages per second. In addition, Vuvuzela protects users even when adoption of the system is small; thus, an adversary's ability to detect communication patterns is limited despite the number of users in the system.

This thesis uses Vuvuzela as a basis for both the multi-party private messaging models as well as developing a distributed private messaging system, Stadium [45].

2.1 Vuvuzela

For the purposes of this work, the author briefly outlines the Vuvuzela system and highlights (key terms presented emphasized with *italics*) the important parts that are relevant to the discussion in this project. The author encourages the reader to read Vuvuzela [46] in order to gain a better understanding of the system that this work extends.

At a high level, a *client* first submits a message to an entry server in a given *round* and then the Vuvuzela system proceeds on the server-side. Vuvuzela takes the user messages and routes them through a *chain* of *servers*, of which at least one is honest. The user messages are deposited in *mailboxes* or *dead drops* which are simply buckets for exchange such that the messages can be swapped and routed back to the recipient. In order to protect the communication patterns of the users, Vuvuzela provides three main functionalities. First, Vuvuzela uses a *mixnet* in order to shuffle messages in the network so that the ordering of messages is not associated with users. Second, Vuvuzela uses constant-bandwidth protocols which ensures that clients are sending messages at a same rate with encrypted messages of the same size so that a message appears the same to any eavesdropping adversary. Third, Vuvuzela incorporates *cover traffic* or *noise* so as to ensure that an adversarial server who has the ability to tamper with the network gleans negligible information about the communication patterns in a given round.

It is important to note, however, that a persistent adversary can, over multiple rounds of the Vuvuzela system, build evidence of communication patterns. To address this, Vuvuzela incorporates *differential privacy* [16, 17] which is formalization of "plausible deniability". Simplistically, given the observable outcome of a communication round a user, Alice, can claim that her actions could just have likely been represented been some other possible action with a near equivalent probability. The composition of differential privacy over multiple rounds allows Alice to construct a *cover story* that could represent Alice's long term actions that may be different from her true actions.

Vuvuzela is split into two protocols: a *dialing* protocol and a *conversation* protocol. The dialing protocol allows users to initiate conversations with one another and relies on *invitation messages*. The conversation protocol is the mainstay of the Vuvuzela system and provides a mechanism for users who are already in a conversation with one another to exchange messages. Both proceed in a similar manner to the high level introduction presented above.

2.2 Goals

The goal of this research work is to extend Vuvuzela to support multi-party private messaging as well as improve the scale and cost-efficiency of Vuvuzela with a distributed private messaging system.

With regard to *privacy*, a strong adversary, one who can observe and tamper with the entire network except for some stated number of honest servers, should not be able to distinguish between one possible user communication behavior and another valid communication behavior, even after interfering with the system. Thus, the system we investigate must be resilient to an adversary temporarily blocking network traffic from a client, altering messages passed within the system, or accessing stateful or stateless information on a server.

For formality, we provide the definition of differential privacy and refer the reader to [17] for additional reading. This definition will be used later in our analyses.

Definition 2.2.1 (Differential Privacy). *A randomized algorithm M is (ε, δ)-differentially private for adjacent pairs of inputs x and y if, for all sets of outcomes S,* $\Pr[M(x) \in S] \leq e^{\varepsilon} \cdot \Pr[M(y) \in S] + \delta$

For brevity, the threat model is not described in this work but is largely identical to that of Vuvuzela's threat model [46]. A slightly different model is presented with respect to the distributed private messaging system, Stadium [45], and the reader is encouraged to view those works for a more holistic picture.

2.2.1 Multi-party Private Messaging Goals

We briefly list some additional privacy goals associated with the multi-party or group private messaging framework. For our purposes a group consists of at least 2 users. In particular, we wish to obfuscate the following observables:

- An adversary should not be able to determine whether a given user is part of a group or not.
- An adversary should not be able to determine the size of a group.
- An adversary should not be able to determine the number of groups a user is part of.
- An adversary should not be able to determine whether or not a user has sent a message to or received a message from a group.

In addition, we list some design goals we wish to keep in mind when comparing multi-party private messaging models.

- **Latency** - we should optimize for less latency between when any user in a group sends a message and a all other members receive it.
- **Efficiency** - we aim to keep communication overhead and network bandwidth to a minimum and limit cover traffic or noise in the system.
- **Integration** - we aim to provide simple integration and extension of the Vuvuzela system with little added complexity.

2.2.2 Distributed Private Messaging Goals

We briefly list some additional goals associated with the distributed private messaging framework.

- **Scalability** - we aim to support more users in a distributed system.
- **Cost-efficiency** - we strive to reduce the cost barrier that is required for an individual to run a server in the system.
- **Simple Implementation** - our goal is to have a simple implementation of this new system that also provides an easy interface for clients.

With respect to distributed private messaging, this work largely concerns the implementation aspects and focuses on the author's contributions to the collaboratively developed system.

Chapter 3

Multi-party Private Messaging

Multi-party or group private messaging is a natural but non-trivial extension to bidirectional communication models. Multi-party messaging is common in insecure systems such as Internet Relay Chat (IRC) systems or Group Texting. There are some schemes that provide end-to-end encyption and can scale to large numbers of users such as Off-the-record (OTR) messaging [6, 26], TextSecure [44], Telegram [28], and Whatsapp [43]. Though such systems may offer end-to-end encryption of messages or be served over a Tor-like network, methods such as traffic analysis or packet inspection and modification can undermine the security of the communication. This project seeks to provide private messaging which is resistant to such traffic analysis or packet alteration and provide the communicating parties plausible deniability that they were engaged in a conversation. The extension is non-trivial because cryptographic primitives are designed largely for two-party communication [22] and adding more users into a given conversation provides an adversary more degrees of freedom in designing an attack.

The contribution of this thesis is to present an extension to Vuvuzela that would allow it to support multi-party communication and analyze the cost and overhead of such a system. The author provides a description of a multi-party Vuvuzela that could serve as a basis for implementation and evaluation.

3.1 Security goals

Informally we provide the following security guarantee. For some given user, let's say Alice, an adversary should not be able to distinguish between any of Alice's possible communication patterns for many message exchanges. In particular, with a multi-party messaging framework we wish to conceal not only whether Alice is communicating and whom she is speaking with at a given point but also how large a group she is communicating with if any at all. A more formal definition of this security goal is provided in Vuvuzela [46].

In addition, a user should be able to take part in multiple groups of varying sizes and still should be equally protected.

3.2 Designs and Analyses

This thesis presents three possible approaches to solving the multi-party communication problem, and discusses the implications of each technique. The first is a broadcast based approach where each party maintains the bidirectional model of Vuvuzela but instead communicates with all other parties in the group. The second is a chaining approach in which a group is chained together in a ring and messages are forwarded along. The third is a more intricate approach involving a shared-mailbox between each of the parties in the group.

To simplify our discussion of the designs and presentation of the analysis we introduce several terms and notation. A *group*, g, consists of multiple unique users and the upper bound for a group size in a system is given by G. We will use users Alice, Bob, Charles, Dan and Frank as default user names and Eve as an adversary name.

3.2.1 Broadcast Model

Design

The Broadcast Model as depicted in Figure 3-1 is a fairly simple solution to the multi-party messaging problem. If Alice wishes to form a group with Bob, Charles, and Dan, then she would first use a modified Vuvuzela dialing protocol to send invitations to Bob, Charles and Dan. This modified dialing protocol allows for a list of users in the group such that in Bob's invitation he knows that Charles and Dan are also part of the group as well. Conversation rounds are similar to that of Vuvuzela except when sending messages to a group, now Alice must send independently to Bob, Charles, and Dan through two distinct Vuvuzela conversations. Alice can now form an two member group with just Frank and communicate in a similar manner.

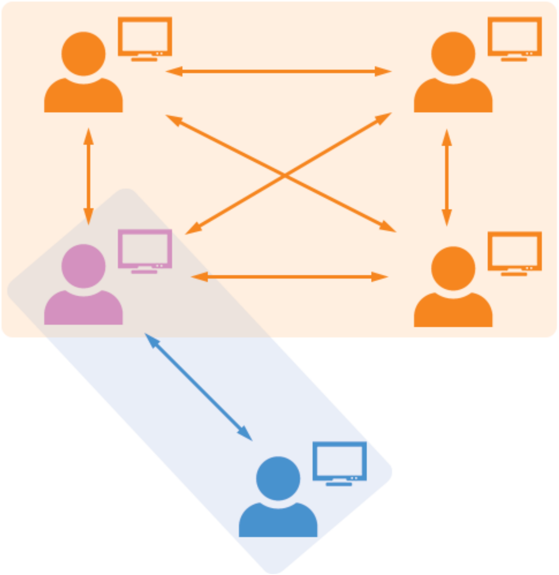

Figure 3-1: Broadcast Model Interaction Schematic. Orange is a group of 4 users and Blue is a group of 2 with a single user overlapped within groups. The bidirectional arrows indicate a standard Vuvuzela two-party protocol.

Analysis

The security of this design is underpinned by the underlying Vuvuzela protocol in place. First, with the dialing protocol is extended from Vuvuzela now to include a list of users (such as a list of public keys of the users) in the group. Vuvuzela already hides the observable variables in the dialing protocol by using fake invitations, a mixnet, and adding cover traffic; thus, the extended dialing protocol is also secure.

With regard to the conversations, Vuvuzela already provides differential privacy to Alice's communication and so she can communicate with Bob, Charles, and Dan at the same time and independently without providing any link that Bob, Charles, and Dan are part of the four-party group. In the reverse style, when Alice receives a message from Bob, Charles, or Dan she is also protected from any information leakage. Finally, when Alice instantiates a conversation with Frank, this is yet another standard Vuvuzela protocol and so no information is leaked. Ultimately, to an adversary such as Eve, the only difference for the this Broadcast Model compared to the original Vuvuzela is that all invitation messages are now larger.

Discussion

In this Broadcast Model, Alice may have to wait multiple rounds in order to send messages to all participants in the group due to the fact that Alice is limited in the number of messages she can send in a given round (in the original Vuvuzela a client sends only 1 message per round, which is what is used our discussions). In particular, for a group of size G, we expect the latency for a message to propagate to all clients to be $O(G)$. Alice must also handle all the delivery of messages herself by starting a Vuvuzela protocol with the other $|g| - 1$ members, where $|g| \leq G$ is the number of members in her group g. Due to this, the client communication complexity grows $O(G)$.

Thus the Broadcast Model adds both latency and places a higher communication burden on a single user when sending messages. However, the communication burden is amortized across all users in a group when considering both sending and receiving messages and each user equally likely to send messages. In addition, the invitation download batches consume more bandwidth as the messages are of larger sizes now.

3.2.2 Chaining Model

Design

The Chaining Model or a Ring Model as depicted in Figure 3-2 aims to reduce the communication burden that Alice must face when sending messages. If Alice wishes to form a group with Bob, Charles, and Dan, Alice will use a modified Vuvuzela dialing protocol to send an invitation to just one of the other members. In particular, a cycle or ring is formed $A \rightarrow B \rightarrow C \rightarrow D \rightarrow A$ (where A is Alice, B is Bob, C is Charles, and D is Dan), such that messages are forwarded along the cycle. Bob receives Alice invitation message and sends one to Charles with the chain included and so on. Thus, once Dan has received the invitation he need not forward it to Alice as she was the head of the chain or cycle, in this case we call the originator of the group, the *leader*. When any of them wish to send a message, they will simply use the established Vuvuzela conversation protocol with the next member of the chain and forward along messages they receive to the next member on the chain until the original sender is reached. Alice can also form a two member group with Frank by sending a chain of $A \rightarrow F \rightarrow A$.

Analysis

The security of this design follows from the original Vuvuzela protocol. First, the dialing protocol is extended from Vuvuzela to allow for additional data in the invitation message, in particular the members of the group in a chained manner (such as a ordered list of public keys of the users). Vuvuzela already hides the observable variables in the dialing protocol by using fake invitations, a mixnet, and adding cover traffic; thus, the extended dialing protocol is also secure.

The conversation scheme is also similar to the standard Vuvuzela in that each link in the chain is merely a Vuvuzela protocol and the fact that each user is chained reveals no additional information. Messages now however must contain information about the originator or a forward-count such that the message is forwarded the appropriate number of times. As Alice is protected by differential privacy her communication

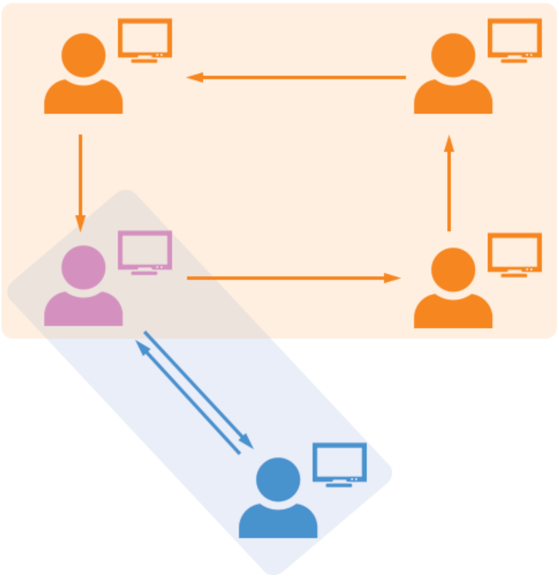

Figure 3-2: Chaining Model Interaction Schematic. Orange is a group of 4 users and Blue is a group of 2 with a single user overlapped within groups. The unidirectional arrows indicate a standard Vuvuzela two-party protocol, however users only forward messages in a single direction and receive in a single direction.

with Bob who in turn has differential privacy when communicating with Charles and so on. An adversary, Eve, cannot discern that Bob is part of the group also including Alice, Charles, and Dan as Bob is able to deny receiving messages from Alice and sending messages to Charles or Dan. To an adversary, this is merely multiple Vuvuzela protocols occurring where sizes of groups nor members of groups cannot be determined. Ultimately, the only difference to an adversary is that invitation messages are larger for all users.

Discussion

In the Chaining Model, Alice has to wait multiple rounds to receive messages from a participant in her group. In general, for a group g of size $|g| \leq G$, on average Alice must now wait $|g|/2$ rounds to receive a message or in general $O(G)$ in terms of receiving message latency. While she can send messages in a single round, she must wait $|g|-1$ rounds to ensure that each member of the group has received her message. Finally, in terms of communication cost, Alice's message only costs her a single round of message sending and forwarding is similar.

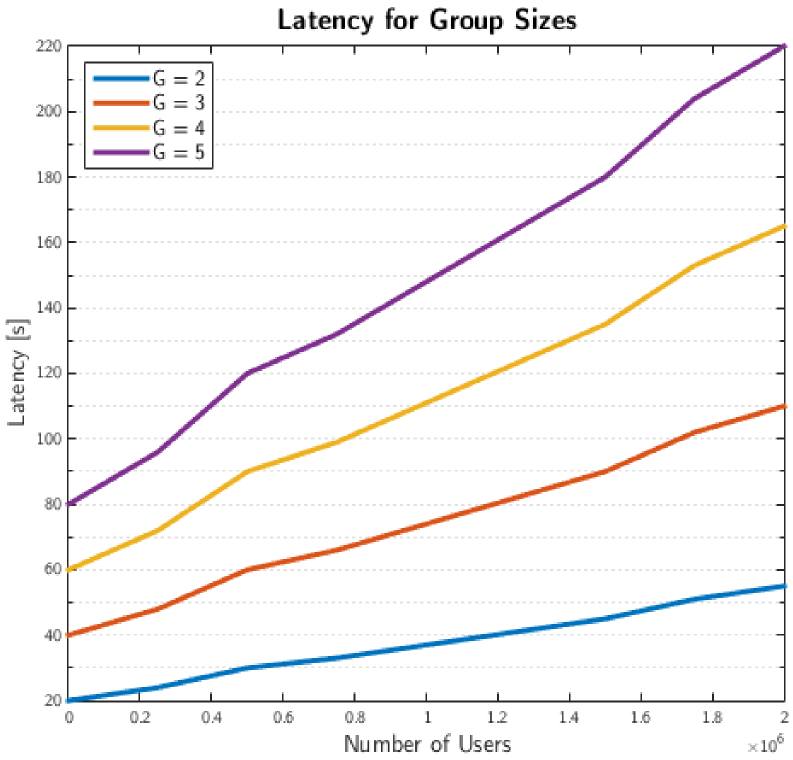

Figure 3-3: Chaining Model Latency for Various Group Sizes. Increasing the bound on the group size G increases the latency for all users in the group to have received a message from any other member. The latency numbers are extrapolated from Vuvuzela's experimental results for varying the number of users the system supports.

Figure 3-3 shows the how the latency grows the number of users in the system with various group sizes. In particular, the latency represent the amount of time required for all members of a group to successfully propagate and receive any messages that were sent. The numbers are extrapolated from the Vuvuzela experimental data. There is a clear tradeoff between handling larger group sizes and the end-to-end latency of the system. For instance, the original two-party Vuvuzela system requires 37 seconds to handle 1 million users, where as the extended systems require 74, 111 and 148 seconds to support three-party, four-party, and five-party private messaging systems respectively.

Thus the Chaining Model adds latency but does a better job of distributing the communication burden across all users even if users do not send messages with the same probability. In addition, the invitation download batches consume more bandwidth as the messages are of larger sizes now. Note that we could add some bidirectionality to model by allowing messages to be forwarded in both directions, however, for simplicity this approach is not presented.

3.2.3 Shared Mailbox Model

Design

The Shared Mailbox Model as depicted in Figure 3-4 is a more intricate model that aims to reduce the latency that exists in both the Broadcast and Chaining models. If Alice wishes to form a group with Bob, Charles and Dan, Alice will use a modified Vuvuzela dialing protocol to send an invitation to both Bob, Charles and Dan. For each group, there will be a designated *leader* who starts the group conversation by sending a invitation message with a shared secret key s to each other user as part of the encrypted invitation message along with a list of users in the group. Thereafter they communicate using a shared dead drop or mailbox location as determined by the original Vuvuzela as $b = H(s, r)$ where b is the dead drop location, H is a standard cryptographic hash function, s is the shared secret key, and r is the round number. For a group g of size $|g| \leq G$, a client will submit $|g| - 1$ messages into the system in order to send a message. The mailboxes themselves perform the exchanges and clients then download the appropriate number of messages. In this system, however, in order to obfuscate the sizes of the groups that individuals are in, we add additional cover traffic such that access counts for 1 to G are covered with noise.

Analysis

The security of this design follows from the original Vuvuzela protocol with some minor adjustments. The *leader* is required to be trusted in the group so the shared secret, s, is common to all parties in the conversation. This isn't a large departure from the original protocol in which a conversation is initiated when the recipient party acknowledges the invitation sender's request. Vuvuzela hides the observable variables in the dialing protocol by using fake invitations, a mixnet, and adding cover traffic;

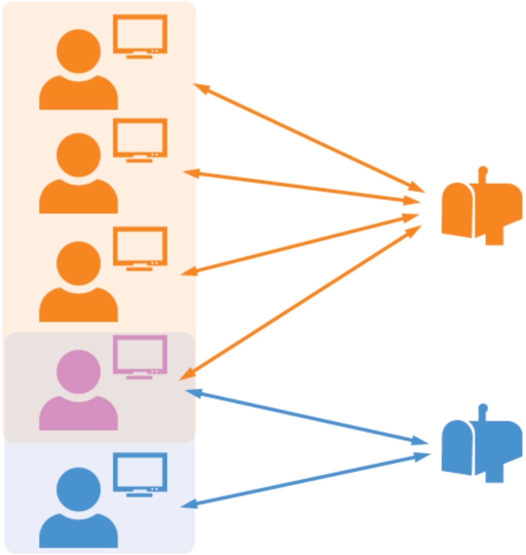

Figure 3-4: Shared Mailbox Model Interaction Schematic. Orange is a group of 4 users and Blue is a group of 2 with a single user overlapped within groups. In this model the modified Vuvuzela protocol allows more than 2 users to access a mailbox thus access counts for a mailbox vary from 1 to G. Noise is added so that the access counts are sufficiently protected against traffic analysis and adversarial tampering.

thus, the extended dialing protocol is also secure.

The conversation protocol now requires further analysis as the noise messages added in the system add further complexity to the design. We wish to ensure a differentially private messaging system and so we identify and minimize the number of observable variables. Similar to the original protocol, we now have access counts to the size of G and so we must consider observable access counts of $m_1, m_2, \ldots, m_G$. Table 3.1 details how these observable variables change with $G = 3$. From the tabular formulation we can see that $\max |\Delta m_i| = G - i + 1$. for each i.

We now show that by adding noise to the m_i with a Laplace distribution with parameters μ, b capped below 0 provides differential privacy.

Theorem 3.2.1. *Consider the algorithm M that adds noise $\lceil \max\left(0, \text{Laplace}(\frac{\mu}{i}, \frac{b}{i})\right) \rceil$ to m_i. Then M is (ε, δ)-differentially private with respect to changes of up to $G-i+1$ in m_i, for $\varepsilon = \frac{G(G+1)(G+2)}{6b}$ and $\delta = \sum_{i=1}^{G} \frac{1}{2} \exp\left(\frac{G-i+1-\mu/i}{b/i}\right)$.*

Proof. We use Lemma 3 from [46] to prove this claim. If we consider an algorithm $M(x)$ that add noise $N \sim \lceil \max(0, \text{Laplace}(\mu, b)) \rceil$ to value $x \geq 0$, then M is (ε, δ)

		Alice's real action			
		Idle	**Conv with b,c**	**Conv with b** **Conv with c**	**Conv with d**
Alice's Cover Story	**Idle**	0,0,0	-3,0,+1	-3,+2,0	-1,+1,0
	Conv with b,c	+3,0,-1	0,0,0	0,+2,-1	+2,+1,-1
	Conv with b, Conv with c	+3,-2,0	0,-2,+1	0,0,0	+2,-1,0
	Conv with d	+1,-1,0	-2,-1,+1	-2,+1,0	0,0,0
	Conv with x,y	+3,0,-1	0,0,0	0,+2,-1	+2,+1,-1
	Conv with x, Conv with y	+3,-2,0	0,-2,+1	0,0,0	+2,-1,0
	Conv with z	+1,-1,0	-2,-1,+1	-2,+1,0	0,0,0

Table 3.1: Difference in Access Counts for Group Size 3. Each cell contains the tuple $(\Delta m_1, \Delta m_2, \Delta m_3)$ which is the difference in access counts of m_i for $i \in \{1, 2, 3\}$ between Alice's real action and cover story. b, c, and d are other users currently conversing with Alice, x, y, and z are users not conversing with Alice.

differentially private with respect to changes of up to t in x for parameters $\varepsilon = t/b$ and $\delta = \frac{1}{2}\exp\left(\frac{t-\mu}{b}\right)$.

We have that an adversary can observe all m_i and that we add noise separately to each of them with Laplace distribution parameters of $(\mu/i, b/i)$ for m_i. $|m_i|$ changes by at most $G - i + 1$ when we modify the action of one user in one round. Thus, the overall scheme provides composed privacy parameters of

$$\varepsilon = \sum_{i=1}^{G} \frac{(G - i + 1) \times i}{b} = \frac{G(G+1)(G+2)}{6b} \tag{3.1}$$

and

$$\delta = \sum_{i=1}^{G} \frac{1}{2} \exp\left(\frac{G - i + 1 - \mu/i}{b/i}\right) \tag{3.2}$$

□

We then wish to know how this system performs under multiple rounds of communication where the adversary can perturb each round. This scenario is known as adaptive composition in differential privacy. If we denote k as the number of rounds of communication.

Theorem 3.2.2. *Consider the algorithm M that adds noise $\left\lceil \max\left(0, \text{Laplace}(\frac{\mu}{i}, \frac{b}{i})\right)\right\rceil$ to m_i* ***over k Vuvuzela rounds****. Then M is (ε', δ')-differentially private with respect to the actions of one user in these rounds with the parameters.*

$$\varepsilon' = \sqrt{2k\ln(1/d)}\varepsilon + k\varepsilon\left(e^{\varepsilon} - 1\right) \quad \textit{and} \quad \delta' = k\delta + d$$

for any $d > 0$, where ε and δ are given in Theorem 3.2.1.

Proof. Direct application of Theorem 3.20 in [18]. □

Discussion

In the Shared Mailbox Model, Alice no longer has to wait only one round to receive messages from any participant in her group. However, in terms of communication cost, for a group g of size $|g| \leq G$, Alice must send $|g| - 1$ messages increasing the network cost in this extended Vuvuzela system.

We can then see how the differential privacy parameters perform for various values of G in the shared mailbox model in Figure 3-5.

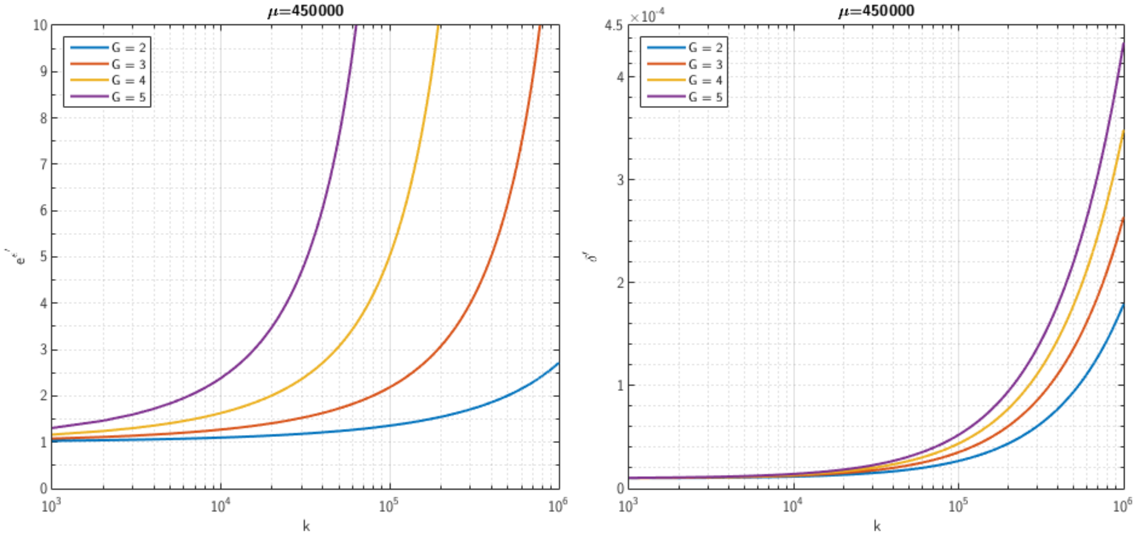

Figure 3-5: Shared Mailbox Model Differential Privacy Parameters for Various Group Sizes. Increasing the bound on the group size G decreases the plausible deniability parameters (left) and also decreases the zero probability circumstance (right). Number of rounds, k, varies from a thousand to a million rounds.

The parameter ε' describes the *plausible deniability* that Alice has and quantifies describes the multiplicative change in the probability of each outcome given Alice's actions. Specifically, if Eve suspects with $p_{prior} = 50\%$ that Alice is part of some group g, then using Bayes' Law Eve's posterior belief that Alice is part of that group is $p_{posterior} = 67\%$ for $\varepsilon' = \ln 2$ and $p_{posterior} = 75\%$ for $\varepsilon' = \ln 3$. This application is described in further detail in [18].

The parameter δ' covers the case where events have *zero* probability under some actions and not under others. For instance, if there are zero access counts to the variable m_i then we know that no group of size $i \leq G$ exists over the k rounds. Keeping a low δ' is important to ensure that those events are unlikely.

3.3 Discussion

Ultimately, each of the models for a multi-party private messaging system presented above has different tradeoffs. For the Broadcast Model the network bandwidth is large for a single user as each client must broadcast their messages to all other parties when attempting to send a message. Furthermore, this method adds latency if all else is kept the same since a user must wait multiple rounds to dispatch its queued messages. For the Chaining Model the network bandwidth is amortized overall since when a client sends a message the other clients will participate in forwarding that message and so a single user need not consume their own network bandwidth for sending messages. However, on average each user will still have to participate in forwarding messages of other clients within their group making the overall bandwidth the same as the Broadcast Model. The latency of this system is necessarily large since a user must wait a number of rounds for a message to be sent around the chain which scales with the number of users in the group. Finally, the Shared Mailbox Model attempts to reconcile the latency issue by introducing more noise into the system. Users may communicate with all others in their group within a single round, however, the network bandwidth remains the same overall since each user must receive a copy of the message being sent.

A challenge posed is to come up with a model that reduces the network bandwidth required in the system while still protecting privacy. The problem is that each user should still receive a copy of the message sent in a group conversation, and attempts to reduce this replication must also be concerned with possible linking of two or more users together within a group.

Chapter 4

Distributed Private Messaging

Distributed Private Messaging is an attempt to improve the efficiency of current private messaging frameworks. In particular, Vuvuzela can be expensive for a single server as all messages in the system must pass through its server. A distributed system aims to distribute the load that servers must handle as part of the overall system, in this case the number of messages handled. The consequence is that it allows smaller machines to be servers or equivalently cut the cost of running a server and also allow the system to scale to more users.

Stadium is a distributed metadata-private messaging system addresses these concerns by scaling the work of the private messaging system across multiple servers [45]. The author contributed to Stadium with some design choices and largely with implementation and evaluation of the system. As part of this thesis, the author describes the contributions made and where it fits into the distributed private messaging system. For a complete understanding of the system, however, the reader is encouraged to read [45].

4.1 Overview

Stadium introduces a verifiable parallel mixnet design. The mixnet design is similar to Vuvuzela in that messages in the system are mixed together by a network of servers deposited in dead drops or mailboxes and then returned back to the clients. The parallel mixnet [23] aspect arises in that multiple paths through the network can exist as clients choose random *chains* of servers and the processing of the messages in a round is performed in parallel. Finally, the verifiable aspect of Stadium occurs in both the distribution of messages from one server to another server as well as when shuffling messages [7] to hide any information from the original ordering of messages.

Stadium assumes some fraction of honest servers such that, with high probability, a selected *chain* contains at least one honest server.

4.2 Implementation

Stadium is implemented largely in Go in under 3000 lines of code and also contains C++ code for cryptographic operations such as the verifiable shuffling [4] which uses the NTL library [42] and python code for handling Merkle Tree operations.

The author is responsible for implementation of parts of the the communication pipeline, verifiable distribution, noise generation, and prover-verifier shuffling testing framework in the Stadium system.

4.2.1 Communication Pipeline

The communication pipeline consists of several phases. Noise is added into the system and clients submit messages to servers. The messages are then mixed and verified in a first mixing phase, distributed and verified, mixed and verified in a second mixing phase, and finally deposited into mailboxes. The messages then are returned back to the clients through the chain of servers. A server processes message *batches* which are a list of encrypted messages and associated onion-encrypted metadata as well as message *parcels* which are subsets of the message batches delivered to the next server in the chain and distribution pipeline.

Stadium's Go implementation includes a server whose responsibility is to handle these multiple phases. The author is responsible for implementing the techniques used for passing message batches and parcels through the system. In addition, the author implemented the functionality to return messages back to the user by first performing exchanging messages at the mailboxes followed by using the state of the servers to delineate the path back to the client.

4.2.2 Noise Generation

Noise plays an important role in the system by obfuscating the access counts of mailboxes and providing the necessary cover traffic such that an adversary cannot isolate and attack user messages. The author implemented the functionality to generate noise for both single and double access counts by producing fake messages to propagate through the system. In order to provide the necessary privacy guarantees, the number of noise messages is important and so the author implemented a sampling method for a Laplace distribution to obtain a random number of noise messages that fits the necessary privacy requirements. The Noise Generation procedure is implemented entirely in Go.

4.2.3 Merkle Tree Verification

Verifiable distribution is an important aspect of Stadium in order to protect against malicious servers. In particular, a server may attempt to undermine the privacy of the system or an individual user by removing, altering, or duplicating messages in the system. Servers audit other servers during the distribution process in order to protect against the aforementioned malicious behavior. In order to do so, each server maintains a Merkle tree of the messages passing through the server. Thus after a distribution phase, a recipient server can check with each of the forwarding servers that all messages received are indeed from some forwarding server and furthermore that all messages sent have indeed been received by some receiving server. Merkle trees allow for quick verification of these large message batches in an efficient manner.

Stadium's implementation uses an internal python server to construct and store Merkle trees and the Go server provides a wrapper to access this Merkle server to construct trees, obtain Merkle roots, and verify paths.

An extension to this approach is to have forwarding servers send the hash value of the message parcel (a subset of the message batch it uses) to the recipient server. The recipient servers then simply need to ensure that the hash values are consistent with their received message parcels.

4.2.4 Prover and Verifier Shuffling Testing Framework

The author is responsible in part for developing a testing framework used for determining the efficiency of the prover and verifier components of the verifiable shuffling procedure. This involved writing python code to launch C++ servers that would perform the interactive prover-verifier procedure.

4.3 Evaluation

Stadium provides a more scalable and efficient alternative to Vuvuzela as determined by a series of evaluations. The author is responsible in part for benchmarking various modules within the system as well as testing the Stadium system end-to-end. In order to generate the necessary performance numbers that would represent a realistic use of Stadium, the author evaluated the system using Amazon Web Services (AWS) Elastic Cloud Computing (EC2) instances [41]. The machines we deploy run 64-bit Linux kernel with an Ubuntu 14.04 LTS operating system and Go 1.6. In all our experiments, all machines run on servers all in the same availability zone; in a realistic system these servers will be running across various data centers and geographic regions. However, since network latency has less effect on the overall latency of Stadium as it is computational bound, we do not consider the additional network latency effect in our analysis.

4.3.1 Prover and Verifier Experiments

The verifiable shuffling procedure is a heavy computational task that serves as the bottleneck of the Stadium system at small to moderate scales. The servers are computational bound and so by benchmarking the computational speeds of the prover and verifier in the shuffling procedure we can obtain an idea of the latency and performance of the Stadium system. The number of provers and verifiers chosen in each of the experiments reflects the size of the chain in the configuration and subsequently the number of prover or verifier instances that need to be generated. A parameter sweep is used to determine optimal configuration sizes and the resulting latency of the end-to-end system.

	Number of Messages		
Provers	**32768**	**65536**	**131072**
12	15.81	33.44	86.77
10	13.70	28.13	72.49
8	10.85	23.53	59.80
6	8.477	18.15	45.36
4	6.991	13.38	32.76
3	5.975	11.62	26.57
1	4.674	9.252	18.49

Table 4.1: Prover Computational Benchmark. Each cell shows the maximum number of seconds required for a prover to process the corresponding number of messages with the associated number of provers on a single `c4.8xlarge` machine.

In Table 4.1 we have the results of an experiment varying the number of prover instances on a single `c4.8xlarge` Amazon EC2 VM with 36 Intel Xeon E5-2666v3 cores, 60 GB of RAM, and 10 Gbps of network bandwidth. The verifier is an identical 36-core machine that participates in the other half of the interactive proof and verification. In addition, we vary the number of messages involved in the interactive proof. In Figure 4-1, we can see the general trend that as we increase the number of provers on a single machine they compete for computational resources and time increases. Similarly, as the increase the number of messages as part of the interactive proof we can see that the duration increases. Both of these trends are as expected. These results are then used to design a configuration with a target latency.

In Table 4.2 we have the results of an experiment varying the number of verifier instances on a single `c4.8xlarge` Amazon EC2 VM with 36 Intel Xeon E5-2666v3 cores, 60 GB of RAM, and 10 Gbps of network bandwidth. The prover instances, whose runtimes are not presented above, are 4 different machines: `c4.8xlarge` (36 cores), `c3.8xlarge` (32 cores), `c4.4xlarge` (16 cores), and `c3.4xlarge` (16 cores).

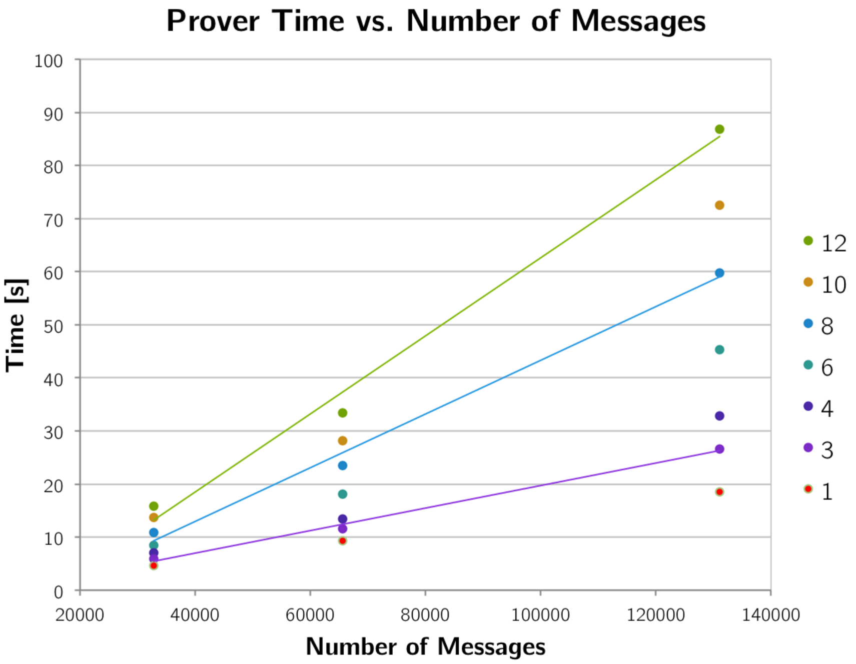

Figure 4-1: Prover Computational Benchmark Graph. The graph shows an increasing trend in the number of seconds required for a prover as the number of messages increases with select prover counts fitted with a linear regression line.

	Number of Messages								
Verifiers	**32768**	**40960**	**49152**	**57344**	**65536**	**81920**	**98304**	**114688**	**131072**
72	13.85	17.45	18.56	24.66	24.52	34.10	38.98	52.64	52.87
60	15.02	–	–	–	21.37	–	–	–	57.37
48	12.08	13.42	16.35	17.34	21.05	28.81	34.54	40.50	45.10
36	10.25	–	–	–	20.33	–	–	–	39.57
24	8.499	–	–	–	16.16	–	–	–	36.89
12	6.678	–	–	–	12.89	–	–	–	25.80

Table 4.2: Verifier Computational Benchmark. Each cell shows the maximum number of seconds required for a verifier to process the corresponding number of messages with the associated number of verifier on a single `c4.8xlarge` machine.

By varying the prover machines, the performance of the verifier is more realistically modeled and performance is improved as the interactive sessions are staggered by a natural jittering since the prover machines perform their computations as differing speeds. In addition, we vary the number of messages involved in the interactive proof. In Figure 4-2, we can see the general trend that as we increase the number of verifiers on a single machine they compete for computational resources and time increases. Similarly, as we increase the number of messages as part of the interactive proof we can see that the duration increases. Both of these trends are as expected. These results are then used to design a configuration with a target latency.

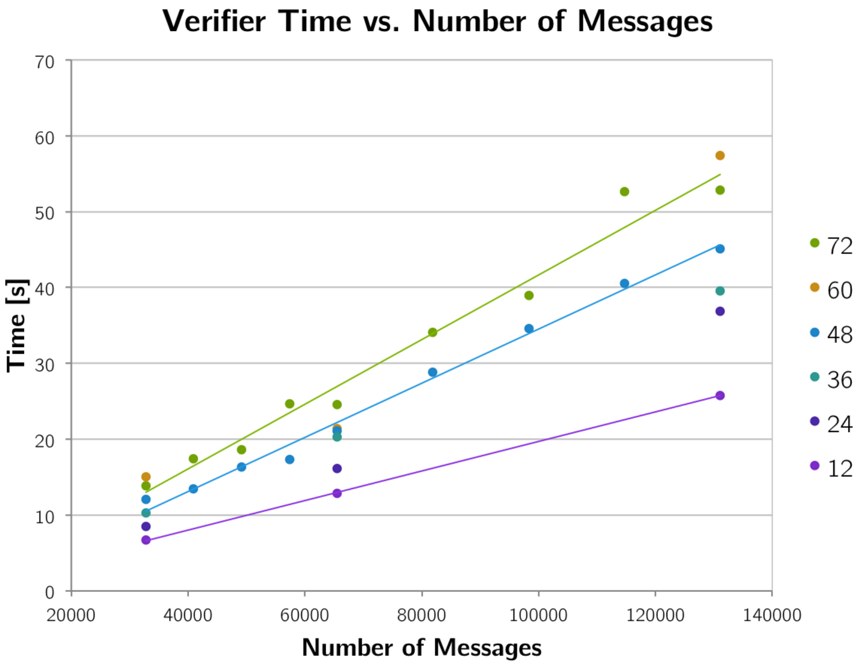

Figure 4-2: Verifier Computational Benchmark Graph. The graph shows an increasing trend in the number of seconds required for a verifier as the number of messages increases with select verifier counts fitted with a linear regression line.

4.3.2 Multicore Experiments

Since the verifiable shuffling procedure is a heavy computational task we examine how adding more cores to the prover and verifier changes the baseline computational time required. We use two `c4.8xlarge` Amazon EC2 VMs with 36 Intel Xeon E5-2666v3 cores.

Cores	Prover Time	Verifier Time
36	21.07	19.51
20	34.56	21.63
10	35.09	21.88
5	57.13	24.67

Table 4.3: Multicore Computational Benchmark. Each cell shows the number of seconds required for a single verifier and single prover to process 150K messages both running on a `c4.8xlarge` machine.

In Table 4.3 we have the results of an experiment varying the number of cores used in the interactive prover-verifier protocol. We use a fixed number of messages at 150K. In Figure 4-3, we can see the general trend that as we increase the number of cores the computational time decreases as expected. However, if we continue to increase the number of cores the tradeoff for computational gain diminishes. Thus a tradeoff between the cost of using a more powerful machine and the latency benefit we achieve is realized.

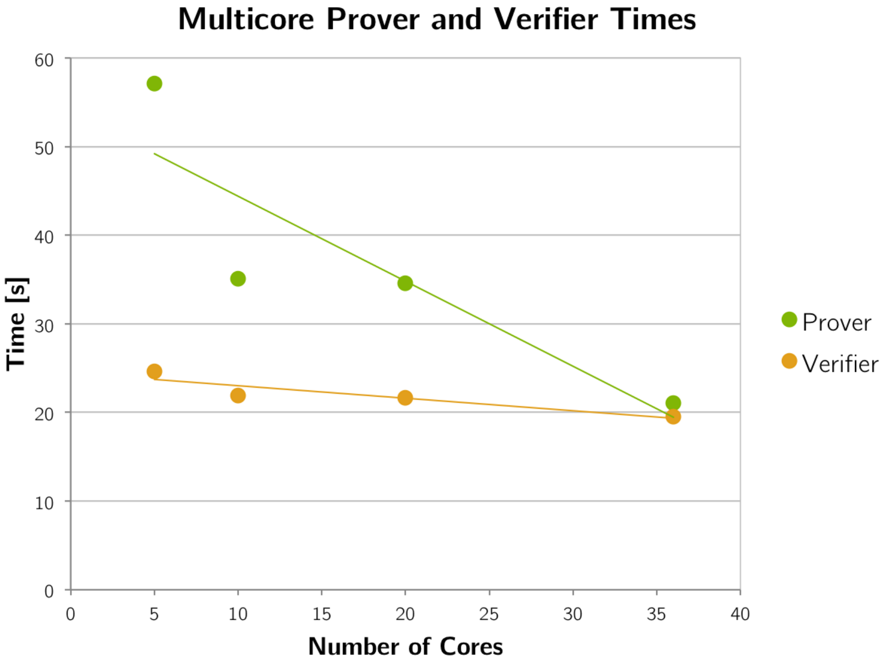

Figure 4-3: Multicore Computational Benchmark Graph. The graph shows a downward trend in the number of seconds required for both a prover and verifier as the number of cores increases. The prover and verifier are both fitted with a linear regression line.

4.4 Discussion

Stadium successfully distributes the server load of Vuvuzela across multiple servers and allows for scaling to more users. The tradeoff however is that latency is increased due to the verifiable procedures and in particular the shuffling that is required. In addition, compared to Vuvuzela each server now handles less of the overall network bandwidth and so the cost of deploying a server is cheaper. This in turn would allow Stadium to be more deployable in a Tor-like setting where “only 5% of relays offer more than 100 Mbps of bandwidth and none offer more than 1 Gbps” [45, 33].

The author’s contributions in implementing the system also include parallelizing some of the communication and verifiable processes. However, the system can be further optimized by parallelizing more of the shuffling procedure. For figures on the scale that Stadium is able to reach and a more detailed explanation of the system the reader is encouraged to look at [45].

Chapter 5

Related Work

We briefly discuss some related work and describe how the multi-party private messaging extension to Vuvuzela and Stadium fit into these contexts.

5.1 Secure Messaging

Secure messaging that offer end-to-end encryption like Off-the-record (OTR) messaging [6, 26], TextSecure [44], Telegram [28], and Whatsapp [43] only encrypt the content of the message. These systems still leak metadata such as which users are communicating and how frequently. For instance, Pond [27] states that "seeks to prevent leaking traffic information against everyone except a global passive attacker". The multi-party private messaging extension to Vuvuzela presented here along with Stadium both protect against metadata even with a global passive adversary.

5.2 Anonymous Communication Systems

Anonymous communication has been around with Chaum's original work on mixnets [8]. However, these previous systems do not protect against traffic analysis and provide scalability at the same time. Mixnet-style systems [8, 14, 19] such as Crowds [35], Freenet [10], Alt.Anonymous.Messages [24], and onion routing [15, 34] can scale to millions of users but are susceptible to a strong adversary [12, 21, 31, 2, 36].

Other systems provide strong security guarantees but do not scale well. Herbivore [37] and Dissent [47] can handle groups of thousands of users whereas Riposte [11] can scale to millions of users but requires limits throughput of the system. Private information retrieval (PIR) systems [9] such as Pynchon Gate [40] are computational expensive requiring quadratic computation in the number of users.

The multi-party private messaging extension to Vuvuzela presented here along with Stadium can protect against traffic analysis and scale well. The multi-party private messaging system is better bounded in its scale by the size of groups permitted.

5.3 Cover Traffic

Mixnet and onion routing systems have used cover traffic or noise to make traffic analysis harder [5, 19, 29]. The multi-party extension to Vuvuzela and Stadium minimize the amount of noise by reducing the number of variables the adversary observes, which allows the noise costs to not dominate the system.

5.4 Differential Privacy

Several systems use differential privacy to provide privacy guarantees. Unlike PINQ [30] and AnoA [3] the systems presented here allow for deviation from protocol as well as composition of privacy over multiple rounds as described by Danezis [13]. However, his work does not attempt to reduce information leakage over hundreds of thousands of rounds.

5.5 Verifiable Mixnets

Stadium incorporates verifiable distribution and shuffling so that audits can be make to identify deviant servers. Verifiable shuffle has been discussed in other privacy-related contexts such as voting schemes [7, 32, 20, 4, 1]. In this work, we apply the verifiable procedures for private messaging.

Chapter 6

Future Work and Conclusions

This project explored the area of private messaging and detailed contributions in multi-party private messaging as well as distributed private messaging. In this last section we first examine some additional work that can be done to improve the private messaging systems discussed and then concludes by describing the overall contribution of this thesis.

6.1 Future Work

First with regard to multi-party private messaging, there are several open questions and development tasks that remain.

- Does there exist a private messaging design that permits messages to be distributed to all parties, but not replicated to the number of users in a group? This would greatly reduce the network cost associated with the proposed multi-party messaging schemes.

- Implementing the proposed models, and in particular the Shared Mailbox Model, is an important step to first evaluate the system and then determine the practical usefulness of group messaging.

Next with regard to distributed private messaging, there are several development tasks that remain.

- Creating a client interface for the system that would allow users to submit messages into the Stadium system. At the moment, servers create user messages for evaluation purposes.

- Deploying the system in a larger configuration. At the moment, the system has only been deployed in a 9 server configuration. To achieve the scale of users and cost-efficiency having the system deployed and evaluated at a more expansive configuration is necessary.

- At the moment, Stadium lacks fault-tolerance properties which make it susceptible to mount a denial of service attack or corrupt the system. To make the system practically useful, one would need to design a more robust Stadium that can withstand these attacks as well as common machine failures.

6.2 Conclusions

In conclusion, private messaging is an important and challenging line of research in computer systems. There is an increasing real-world demand for such systems as metadata of communications, rather than content itself, is more open and susceptible. This thesis project expands on the work of Vuvuzela [46], a private messaging system resistant to traffic analysis, by extending the system to provide the ability for multi-party communication or group conversations as well as distributed private messaging.

This work presented three models for multi-party private messaging and analyzes the tradeoffs of each approach. A key aspect of this contribution is that it easily extends the current Vuvuzela system reducing the need to design an entirely new system. Analysis of the models show that the cost of these multi-party systems grows exponentially with the number of users in the group. Thus, the feasibility of these models is limited to smaller group. Furthermore, the privacy guarantees of these group-based systems is worse for larger group sizes. Ultimately, there is more work that can be done to further improve the efficacy and security of multi-party messaging systems.

This work also discusses contributions made towards a distributed private messaging system. Stadium [45] is a system similar to Vuvuzela that scales to more users and reduces the cost-burden that a single server has to face in the private messaging system by integrating a distributed server-side communication framework. The author's contribution in implementing part of the system as well as benchmarking and evaluating parts of the system paves the way for designing an optimal distributed system. Stadium demonstrates the practical usefulness of private messaging and provides strong privacy guarantees while protecting against more vectors of attack by introducing verifiable procedures. There are several ways to further improve Stadium to make the system more useful to users including fault tolerance and improved client interaction.

In summary, this thesis contributes to the literature of private messaging. The author urges the reader to consider privacy as one of the goals in their own communication behaviors. In addition, the author hopes this technical work will contribute to public discourse in the area of private communications.

Bibliography

[1] Masayuki Abe. Universally verifiable mix-net with verification work independent of the number of mix-servers. In *Advances in CryptologyâĂŤEUROCRYPT'98*, pages 437–447. Springer, 1998.

[2] Adam Back, Ulf Möller, and Anton Stiglic. Traffic analysis attacks and trade-offs in anonymity providing systems. In *Information Hiding*, pages 245–257. Springer, 2001.

[3] Michael Backes, Aniket Kate, Praveen Manoharan, Sebastian Meiser, and Esmaeil Mohammadi. Anoa: A framework for analyzing anonymous communication protocols. In *Computer Security Foundations Symposium (CSF), 2013 IEEE 26th*, pages 163–178. IEEE, 2013.

[4] Stephanie Bayer and Jens Groth. Efficient zero-knowledge argument for correctness of a shuffle. In *Advances in Cryptology–EUROCRYPT 2012*, pages 263–280. Springer, 2012.

[5] Oliver Berthold and Heinrich Langos. Dummy traffic against long term intersection attacks. In *Privacy Enhancing Technologies*, pages 110–128. Springer, 2002.

[6] Nikita Borisov, Ian Goldberg, and Eric Brewer. Off-the-record communication, or, why not to use PGP. In *Proceedings of the 2004 ACM workshop on Privacy in the electronic society*, pages 77–84. ACM, 2004.

[7] Chin-Chen Chang and Wen-Bin Wu. A secure voting system on a public network. *Networks*, 29(2):81–87, 1997.

[8] David L Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–90, 1981.

[9] Benny Chor, Eyal Kushilevitz, Oded Goldreich, and Madhu Sudan. Private information retrieval. *Journal of the ACM (JACM)*, 45(6):965–981, 1998.

[10] Ian Clarke, Oskar Sandberg, Brandon Wiley, and Theodore W Hong. Freenet: A distributed anonymous information storage and retrieval system. In *Designing Privacy Enhancing Technologies*, pages 46–66. Springer, 2001.

[11] Henry Corrigan-Gibbs, Dan Boneh, and David Mazières. Riposte: An anonymous messaging system handling millions of users. In *Security and Privacy (SP), 2015 IEEE Symposium on*, pages 321–338. IEEE, 2015.

[12] George Danezis. Statistical disclosure attacks. In *Security and Privacy in the Age of Uncertainty*, pages 421–426. Springer, 2003.

[13] George Danezis. Measuring anonymity: a few thoughts and a differentially private bound. In *Proceedings of the DIMACS Workshop on Measuring Anonymity*, page 26, 2013.

[14] George Danezis, Roger Dingledine, and Nick Mathewson. Mixminion: Design of a type III anonymous remailer protocol. In *Security and Privacy, 2003. Proceedings. 2003 Symposium on*, pages 2–15. IEEE, 2003.

[15] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. Technical report, DTIC Document, 2004.

[16] Cynthia Dwork. Differential privacy. In *Automata, languages and programming*, pages 1–12. Springer, 2006.

[17] Cynthia Dwork. Differential privacy: A survey of results. In *Theory and applications of models of computation*, pages 1–19. Springer, 2008.

[18] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3-4):211–407, 2014.

[19] Michael J Freedman and Robert Morris. Tarzan: A peer-to-peer anonymizing network layer. In *Proceedings of the 9th ACM conference on Computer and communications security*, pages 193–206. ACM, 2002.

[20] Jun Furukawa and Kazue Sako. An efficient scheme for proving a shuffle. In *Advances in CryptologyâĂŤCRYPTO 2001*, pages 368–387. Springer, 2001.

[21] Yossi Gilad and Amir Herzberg. Spying in the dark: TCP and tor traffic analysis. In *Privacy Enhancing Technologies*, pages 100–119. Springer, 2012.

[22] Ian Goldberg, Berkant Ustaoğlu, Matthew D Van Gundy, and Hao Chen. Multi-party off-the-record messaging. In *Proceedings of the 16th ACM conference on Computer and communications security*, pages 358–368. ACM, 2009.

[23] Philippe Golle and Ari Juels. Parallel mixing. In *Proceedings of the 11th ACM conference on Computer and communications security*, pages 220–226. ACM, 2004.

[24] Google Groups. Alt.Anonymous.Messages. https://groups.google.com/forum/#!forum/alt.anonymous.messages, 2016.

[25] M Hayden. The price of privacy: Re-evaluating the NSA. In *John Hopkins Foreign Affairs Symposium*, 2014.

[26] I. Goldberg and The OTR Development Team. Off-the-record messaging. https://otr.cypherpunks.ca/, 2015.

[27] A. Langley. Pond. https://pond.imperialviolet.org/, 2015.

[28] Telegram Messenger LLP. Telegram. https://telegram.org/, 2016.

[29] Nick Mathewson and Roger Dingledine. Practical traffic analysis: Extending and resisting statistical disclosure. In *Privacy Enhancing Technologies*, pages 17–34. Springer, 2004.

[30] Frank D McSherry. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, pages 19–30. ACM, 2009.

[31] Steven J Murdoch and George Danezis. Low-cost traffic analysis of Tor. In *Security and Privacy, 2005 IEEE Symposium on*, pages 183–195. IEEE, 2005.

[32] C Andrew Neff. A verifiable secret shuffle and its application to e-voting. In *Proceedings of the 8th ACM conference on Computer and Communications Security*, pages 116–125. ACM, 2001.

[33] The Tor Project. Tor Metrics: Advertised bandwidth distribution. https://metrics.torproject.org/advbwdist-perc.html?start=2016-02-15&end=2016-05-15&p=100&p=50, May 2016.

[34] Michael G Reed, Paul F Syverson, and David M Goldschlag. Anonymous connections and onion routing. *Selected Areas in Communications, IEEE Journal on*, 16(4):482–494, 1998.

[35] Michael K Reiter and Aviel D Rubin. Crowds: Anonymity for web transactions. *ACM Transactions on Information and System Security (TISSEC)*, 1(1):66–92, 1998.

[36] Tom Ritter. De-anonymizing Alt.Anonymous.Messages. https://ritter.vg/blog-deanonymizing_amm.html, 2013.

[37] Mark Robson, Milo Polte, Sharad Goel, and Emin Sirer. Herbivore: A scalable and efficient protocol for anonymous communication. Technical report, Cornell University, 2003.

[38] Phillip Rogaway. The Moral Character of Cryptographic Work. Technical report, IACR-Cryptology ePrint Archive, 2015.

[39] A. Rusbridger. The Snowden leaks and the public. *The New York Review of Books*, November 2013.

[40] Len Sassaman, Bram Cohen, and Nick Mathewson. The pynchon gate: A secure method of pseudonymous mail retrieval. In *Proceedings of the 2005 ACM workshop on Privacy in the electronic society*, pages 1–9. ACM, 2005.

[41] Amazon Web Services. Amazon EC2 Instance Types. https://aws.amazon.com/ec2/instance-types/, 2016.

[42] V. Shoup. NTL: A Library for doing Number Theory. http://www.shoup.net/ntl/, 2016.

[43] Open Whisper Systems. Open Whisper Systems partners with WhatsApp to provide end-to-end encryption. https://whispersystems.org/blog/whatsapp/, Nov 2014.

[44] Open Whisper Systems. TextSecure protocol v2. https://github.com/WhisperSystems/TextSecure/wiki/ProtocolV2, 2015.

[45] Nirvan et. al. Tyagi. Stadium: A Distributed Metadata-private Messaging System. 2016.

[46] Jelle Van Den Hooff, David Lazar, Matei Zaharia, and Nickolai Zeldovich. Vuvuzela: Scalable private messaging resistant to traffic analysis. In *Proceedings of the 25th Symposium on Operating Systems Principles*, pages 137–152. ACM, 2015.

[47] David Isaac Wolinsky, Henry Corrigan-Gibbs, Bryan Ford, and Aaron Johnson. Dissent in numbers: Making strong anonymity scale. In *Presented as part of the 10th USENIX Symposium on Operating Systems Design and Implementation (OSDI 12)*, pages 179–182, 2012.