New Lower Bounds on the Complexity of Provably Anonymous Onion Routing

by

Miranda Christ

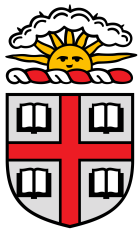
Submitted to the Department of Computer Science
in partial fulfillment of the requirements for the degree of

Bachelor of Science in Computer Science and Mathematics

at

BROWN UNIVERSITY

May 2020



New Lower Bounds on the Complexity of Provably Anonymous Onion Routing

by

Miranda Christ

Submitted to the Department of Computer Science
on May 22, 2020, in partial fulfillment of the
requirements for the degree of
Bachelor of Science in Computer Science and Mathematics

Abstract

In addition to falling under what some believe is a fundamental right to privacy, anonymity is useful for a number of reasons. Imagine, for example, a whistleblower wishing to remain anonymous for his or her safety. Furthermore, anonymity is an essential building block for security systems, many of which assume anonymous channels. Onion routing provides a solution for anonymous communication, where messages are encrypted in layers and relayed via intermediate servers to their recipients. The Trilemma Theorem gives a lower bound on the communication complexity required for anonymous onion routing. In this thesis, we prove tighter lower bounds for an onion routing protocol with security parameter λ anonymous against a passive adversary. First, a protocol anonymous against the passive adversary corrupting a fraction $\frac{1}{f(\lambda)}$ of the servers has a round complexity of $\omega\left(\frac{\log\lambda}{\log f(\lambda)}\right)$. Second, if the maximum number of onions processed at a node in a round is α, a protocol anonymous against the passive adversary corrupting a constant fraction $\frac{1}{f(\lambda)}$ of the servers has a round complexity of $\omega\left(\frac{\log\lambda+\log h}{\log f(\lambda)}+h\right)$, where $h=\frac{\log((1-\kappa)N)}{\log\alpha}$.

Thesis Advisor: Eli Upfal
Title: Professor of Computer Science

Thesis Advisor: Megumi Ando
Title: Ph.D. Student

Thesis Reader: Jeffrey Hoffstein
Title: Professor of Mathematics

Acknowledgments

This thesis would not have been possible without the help of my advisors.

To Megumi, thank you for your mentorship, patience, and encouragement. Thank you also for being so generous with your time while juggling your own thesis, among other things!

To Professor Upfal, thank you for agreeing to advise my thesis and for your expertise and guidance throughout the process.

To Professor Hoffstein, thank you for your enthusiasm in agreeing to be my reader and delving into onion routing.

To my friends and family, thank you for your support from near and far and for making this experience all the more enjoyable.

Contents

Chapter 1

Introduction

1.1 Motivation

The near ubiquity of online communication connects us to those across the world in seconds, but with this convenience comes a cost to privacy, particularly anonymity (concealing who is communicating with whom), as surveillance increases. In addition to falling under what some believe is a fundamental right to privacy, anonymity is useful for a number of reasons; imagine, for example, a whistleblower wishing to remain anonymous for his or her safety. Furthermore, anonymity is an essential building block for security systems, many of which assume anonymous channels.

More formally, a communication protocol is anonymous if an adversary in the standard setting [Gol98] viewing all network traffic and corrupting a fraction of the servers cannot distinguish between inputs; for example, an adversary should not be able to determine whether Alice is talking to Bob, Alice is talking to Carol, or Alice is talking to no one.

The most promising current solution for anonymity is onion routing [Cha81], in which each message is bundled into a cryptographic *onion* with encrypted layers. This onion is passed from the sender to the receiver via intermediate nodes who decrypt along the way. Each intermediate node can decrypt only its layer of the onion. When the onion reaches its final recipient, only one layer remains, which the recipient can decrypt to reveal the message. Since an intermediate node knows only

the previous and next destinations of the onions it passes along, it cannot tell who each onion's original sender and ultimate recipient are. Furthermore, intermediate servers can collect several onions before transmitting them to their next destinations, so the network traffic reveals the set of servers these onions visit next, but not each individual onion's path. Onion routing is advantageous in that it is relatively simple, scalable (it can accommodate many users), and fault-tolerant (its anonymity guarantees are reasonable even if several message packets are dropped). Several systems used in practice are slight variations of onion routing, including Tor [DMS04], used by the Brave web browser [1].

However, Tor is known to be vulnerable to traffic correlation attacks, meaning it is not provably secure [JWJ+13, SEF+17, WSJ+18].

Several provably secure protocols do exist, including Vuvuzela [vdHLZZ15], Stadium [TGL+17], Atom [KCDF17], Π_p and Π_a [ALU18], and $\Pi_{\bowtie}$ [ALU20]. However, these protocols lack efficiency in practice, leading to Tor's prevalence despite its vulnerabilities. Most of these protocols are in the *active adversary* setting (defined in §2.1), where the adversary can see all network traffic and alter the behavior of some number of corrupted parties. There is also a lower bound in the active adversary setting, stating that to obtain anonymity, the expected value of onion transmissions per party must be $\omega(\log \lambda)$, where λ is the security parameter [ALU20].

In this thesis, we consider the passive adversary setting, where the adversary can view all the network traffic and view the actions performed by some number of corrupted parties, but not alter the behavior of these parties. In this passive adversary setting, the provably secure protocol Π_p shows that for every $\epsilon > 0$, anonymity is achievable with $\Omega(\log^{1+\epsilon} \lambda)$ rounds and every party transmitting $\Omega(\log^{1+\epsilon} \lambda)$ onions per round, where λ is the security parameter [ALU18]. Our lower bound presented in Theorem 1 of this thesis shows that Π_p is optimal for the passive adversary corrupting a constant fraction of the parties.

There is an existing lower bound for the passive adversary setting, namely that given in The Trilemma Theorem [DMMK18]. However, their result that each party

[1] https://support.brave.com/hc/en-us/articles/360018121491-What-is-a-Private-Window-with-Tor

needs to send one message packet is trivially true, leaving room for a tighter bound.

1.2 Contributions

In our first result in section 3.1, we prove a lower bound on the round complexity required for anonymity against the passive adversary corrupting a fraction $\frac{1}{f(\lambda)}$ of the parties, that $\omega(\frac{\log \lambda}{\log f(\lambda)})$ rounds are required for anonymity. We obtain this result by bounding the probability that the adversary corrupts all servers in a given onion's routing path. In our second result in section 3.2, we extend this bound for a protocol with a constrained *server load*, meaning for some α, only α onions can meet at a given server at the same time. To obtain this result, we first bound the number of honest servers an onion must pass through to appear that it could have been sent by any party; we then examine the number of rounds required for an onion to pass through this many honest servers. In addition to achieving a tight bound for the passive adversary corrupting a constant fraction of the parties, these results contribute to a more complete set of lower bounds. These lower bounds are useful not only for analyzing the anonymity guarantees provided by existing protocols but also for guiding the design of new protocols.

1.3 Motivating Examples

We introduce several simple examples that should be illustrative of both why we need onion routing and also the complexity associated with it.

Consider a scenario with some n internet users, two of whom are Alice and Bob, and an adversary Eve who is able to observe all network traffic. In other words, Eve can see whenever a packet is sent from one server to another, though she cannot see the contents of the packet. Alice wants to send a message to Bob, but she does not want Eve to know that she is communicating with him. Ideally, Eve should not be able to tell whether Alice is sending a message to Bob, Alice is sending a message to some other user Carol, or Alice is sending no message at all. This inability to

distinguish between scenarios is called anonymity.

This task poses a challenge because if Alice sends her message to Bob directly, even if she encrypts it, Eve will observe her packet traversing the link between their servers and know that they are communicating. We present several simple solutions to this problem with various advantages and drawbacks. Namely, they differ in *round complexity*, *server load*, and *onion cost*. Broadly, round complexity is the number of times each message packet is passed from server to server. Server load is the number of packets each server processes in each round. Onion cost is the expected number of packet transmissions across all servers during a run of the protocol.

1.3.1 Everyone sending to everyone

Assume in this example that the adversary is a network adversary, meaning it can observe all network traffic but not the behavior of any of the parties. In other words, all parties are honest. In this protocol, every party sends a message to every other party in a single round. That is, in addition to Alice sending a message to Bob, she sends a content-less dummy message to every other party. Additionally, every party other than Alice sends a content-less message to every other party. These messages are packaged in such a way that the adversary cannot tell which are content-less and which is Alice's true message to Bob.

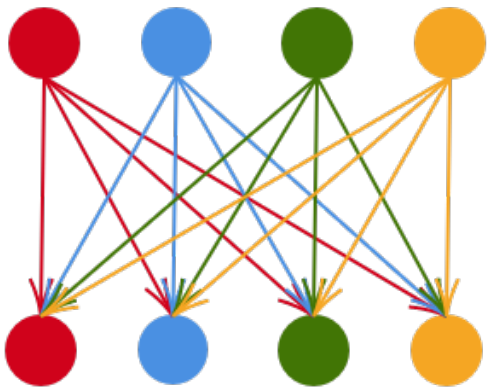


Figure 1-1: Four users, each sending a message to every other user

To anyone observing the network traffic, every party appears to behave identically.

Not only can the adversary not determine who Alice is sending meaningful content to, but she cannot even determine whether Alice (instead of Carol, for example) is the party sending the meaningful message at all.

While this scheme requires no additional servers or rounds, its onion cost is quite high. In order to securely transmit a single meaningful message from Alice to Bob, each of the n parties sends a packet to each of the other n parties.

This example shows that given n parties and a *network adversary* observing all network traffic, anonymity is always achievable with a total of n^2 onion transmissions, even in a single round.

However, this protocol fails against the passive adversary, which can view the actions of some fraction of the servers. If the adversary corrupts the red server, it can see where any message arriving at the red server is coming from. This allows the passive adversary to trace all of the red server's messages back to their senders, breaking anonymity.

1.3.2 Single server

In this protocol, we use a trusted third party S to relay onions between the n users, preventing the attack described above, where the adversary corrupts a recipient. If the adversary corrupts the red user in this example, it sees only that any message arriving at the red server comes from S, instead of from the original sender. Recall that Alice wishes to send a message to Bob. Alice instead sends this message to S, and all other $n-1$ users send a content-less *dummy* (see section 2.1) message to S. Once S receives all of these packets, it then relays them so that Bob receives Alice's message and each other user receives exactly one message.

Eve sees a message packet travel from each user to S, then from S to each user. Observe that the traffic Eve sees is identical for each user, so this protocol reveals no information about Alice and Bob.

The server load of this protocol is n, since S receives packets from every user simultaneously. The round complexity is 2, since each packet travels from its origin to S to its recipient.

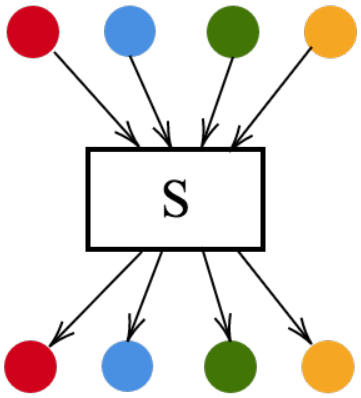


Figure 1-2: Four users, each sending and receiving exactly one message via an intermediate server

While this protocol works with a network adversary observing only the network traffic, it fails in the passive adversary setting, where the adversary can corrupt the central server and see the actions it performs. Since in this thesis we care about the passive adversary setting, this protocol is not sufficient for our purposes.

1.3.3 Many servers

We can solve the above issue by using many central servers $S_1, S_2, ...$ in sequence, ensuring that at least one of them is not corrupted by Eve. Similar to in the above example, each user sends a message to S_1, which relays the messages to S_2, until the final server, which relays them to their respective destinations.

For example, if we want Eve to figure out that Alice is talking to Bob with probability at most $\frac{1}{k}$, and Eve corrupts $\frac{1}{2}$ of the servers, we can safely use $\log_2 k$ servers.

If Eve wants to trace a message packet from origin to recipient, she must corrupt all servers through which it travels. If we use $\log_2 k$ servers, the probability that she corrupts all of them is $\frac{1}{k}$.

While this protocol achieves the desired guarantees against the information leaked to Eve, it has a high cost. Compared to the above example with a single server, the round complexity is much higher ($\log_2 k$, if we want Eve to learn nothing about Alice

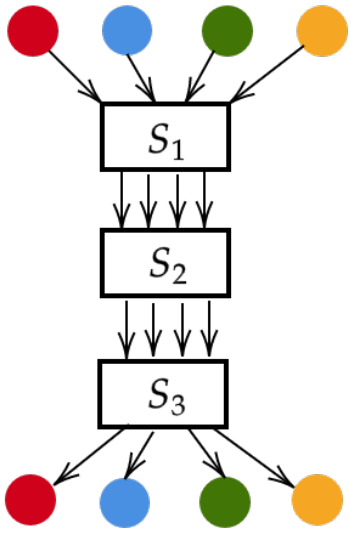


Figure 1-3: Four users, each sending and receiving exactly one message via several intermediate servers

and Bob with probability at least $1 - \frac{1}{k}$), and the server load is the same (n).

1.4 Related Works

There are several known bounds on various cost metrics for onion routing in various adversarial settings, including some of the cost metrics and adversarial settings mentioned above.

The anonymity trilemma [DMMK18] presents bounds on the tradeoff between the server load and round complexity of anonymous communication protocols. In their setting, a run of the protocol involves one user sending a message to another user. This message is routed between servers in communication rounds; each round, it visits one server. The servers can additionally generate noise messages, which help muddle the adversary's view of the genuine message's path through the system. If Carol receives messages from Alice and Bob and passes them along to David and Fred, the adversary cannot tell from the network traffic whether Carol passed Alice's message to David or Fred. However, these noise messages increase the number of transmissions the servers must perform, and a natural question is how many of these additional transmissions are necessary. The anonymity trilemma provides a lower bound for the server load

(the fraction of servers generating noise messages per round) and the number of rounds required for anonymity. Their bound that is most relevant to this thesis is in the passive adversarial setting, where the adversary can observe the traffic across all network links and additionally observe but not control the actions performed by some number c of the servers. More specifically, for an adversary passively compromising c of the parties in a protocol with security parameter λ, number of rounds l, and fraction $\beta \in [0, 1]$ of the parties sending noise messages per communication round, they show that anonymity is impossible if, for $c < l$,

$$2(l - c)\beta < 1 - \frac{1}{poly(\lambda)}$$

or, for $c \geq l$ and $l \in O(1)$,

$$2\beta l < 1 - \frac{1}{poly(\lambda)}$$

where $poly(\lambda)$ denotes any polynomial in λ.

Crucially, if the number of corrupted servers is at least the number of rounds, their bound does not depend on c at all. This leaves room for improvement, and we provide a bound that in this case does depend on c. We wish to provide a tighter bound on the relationship between server load and round complexity; in addition to that, we give separate bounds on each.

Another relevant result is an upper bound in the passive adversarial setting, where an adversary can control a *constant* fraction κ of the parties [ALU18]. Similarly to in this thesis, they use the synchronous communication model and SimpleIO setting. A protocol in the synchronous communication model operates in rounds, where in each round all packets leave their origins simultaneously and arrive at their destinations simultaneously. In the SimpleIO setting, all inputs to the protocol instruct each party to send and receive exactly one message. This setting is fairly standard, and we use it for that reason.

In this paper, the authors prove an upper bound on the number of rounds and onion transmissions sufficient for anonymity by constructing a protocol Π_p. Their construction is fairly simple; each onion is routed through L servers before reaching

its final destination. These servers are chosen independently and uniformly at random. When two honestly formed onions meet at an honest server, these onions get *mixed*, and it becomes difficult to determine which incoming onion corresponds to the outgoing onion. This mixing happens only at honest servers, since the adversary can observe all actions performed by corrupted servers. Intuitively, the more onions each server receives on a given round, the more difficult it becomes for the adversary to determine its origin. Additionally, the more servers a given onion passes through, the more *honest* servers it passes through. The authors show that anonymity is achievable with $\Omega(\log^2 \lambda)$ rounds and every party transmitting $\Omega(\log^2 \lambda)$ onions per round. For any $\epsilon > 0$, their proof also holds with $\Omega(\log^{1+\epsilon} \lambda)$ rounds and every party transmitting $\Omega(\log^{1+\epsilon} \lambda)$ onions per round. The authors also provide bounds for the active adversarial setting, which are improved upon in their more recent paper.

Their most recent result is for the active adversarial setting. In this setting, the adversary is able to view all network traffic, control a constant fraction of the parties, *and* cause these parties to deviate from the protocol. That is, servers that the adversary controls can drop onions as they please. This enables a new kind of attack, where if Eve wishes to determine if Alice is sending a message to a corrupted party Bob, Eve can drop all onions that could have originated from Alice at all servers Eve controls, effectively blocking Alice's onion from reaching Bob. Since Eve controls Bob, Eve can see all messages received by Bob at the end of the protocol. If Bob does not receive an expected message, Eve can infer that that message was likely from Alice and dropped through the course of her attack. They introduce a protocol, $\Pi_{\bowtie}$ (pronounced *Pi-butterfly*), that detects when Eve conducts an attack of this nature and aborts the protocol if so [ALU20]. $\Pi_{\bowtie}$ employs "merging onions" and "checkpoint onions" to this effect. Merging onions ensure that even if the adversary strategically drops onions, they cannot gain information by examining the number of onions each party receives. Checkpoint onions serve as indicators for the honest parties of how many onions remain in the system. Each round, the honest parties tally up the number of checkpoint onions they receive and compare this count to the expected number of checkpoint onions. If at any point an honest party determines via this

count of checkpoint onions that too few onions are in the system, an attack is likely underway, and it sends signals to all other parties to abort the protocol. Once the protocol has been aborted, Eve's attack fails, as all traffic halts and many messages intended for Bob, not just the message sent by Alice, do not reach him.

Via this protocol, Ando, Lysyanskaya, and Upfal prove an upper bound on the complexity (here, we define this as number of message packets) sufficient for anonymity: $\Pi_{\bowtie}$ achieves anonymity against an adversary actively corrupting any fraction $\kappa < \frac{1}{2}$ of the parties, with each honest party transmitting $\gamma_1 \log N \log^{3+\gamma_2} \lambda$ onions, where N is the number of parties, λ is the security parameter, and γ_1, γ_2 are constants. They also prove a near-tight lower bound, that in order to achieve anonymity against an adversary actively corrupting a fraction κ of the parties, every party in the protocol must transmit at least a logarithmic number of onions. In this sense, the authors' construction $\Pi_{\bowtie}$ is near-optimal.

Chapter 2

Preliminaries

Onion encryption schemes draw inspiration from David Chaum's concept of mix networks [Cha81]. In Chaum's model, much like in our example 1.2.2, parties employ an intermediate server to mix their messages, each sending an encrypted message packet to this server, which then forwards it to its recipient. If party A wants to send a message m to party B via this mix network, it encrypts m with B's public key to get m', then encrypts (m', B) with the mixing server's public key. A must include B so that the mixing server knows where to forward the message to. While this model works well for one mixing server, it cannot be directly applied to onion routing, where there are multiple mixing servers in sequence. Since the next recipient is included in each packet, packet size depends on the number of mixing servers that packet will visit. An adversary can then use packet size to determine how long a given packet will remain in the system; the larger a packet, the longer it will remain.

Mixing networks also rely on the fact that all packets exiting a server look the same; if their sizes differ, and these sizes are correlated with the sizes of the incoming packets, the adversary can use this information to trace packets through the system. We therefore need an encryption scheme where the sizes of packets are indistinguishable, regardless of the number of servers in their routing paths. This is exactly what Camenisch and Lysyanskaya propose.

We consider onion routing protocols using onion encryption schemes as defined by Camenisch and Lysyanskaya [CL05]. Their paper presents a cryptographic defi-

nition of an onion encryption scheme that guarantees security in Canetti's universal composability (UC) framework [Can01]. A challenge they tackle is preventing the adversary from distinguishing onions by their sizes. Since each onion must specify where each server in its routing path must send it next, it is natural that each onion's size would depend on the number of rounds for which it exists in the system. This could allow the adversary to determine which onions were created earlier in the protocol run by looking at their sizes. Camenisch and Lysyanskaya describe a set of algorithms that prevent this attack and achieve security. They formalize the three algorithms, G, $\mathsf{FormOnion}$, and $\mathsf{ProcOnion}$. G, given a public parameter, generates the public and private keys for each party. $\mathsf{FormOnion}$, given the public parameter, message, list of parties, and the parties' public keys, generates a sequence of onion layers in probabilistic polynomial time. $\mathsf{ProcOnion}$ "peels" an onion, taking as input an onion, party, and that party's secret key, and yielding as output the "peeled" onion and its next destination. $\mathsf{ProcOnion}$ runs in deterministic polynomial time.

When we talk about an onion routing protocol in this thesis, we mean a communication protocol using Camenisch and Lysyanskaya's set of algorithms G, $\mathsf{FormOnion}$, and $\mathsf{ProcOnion}$.

2.1 Modeling the Problem

Onion routing protocol. We first define onion routing more formally. A *cryptographic onion* is a message packet that is encrypted in layers, with a public key corresponding to each layer. For example, if party P_i wants to send a message m to party P_j via intermediate parties $P_1, ..., P_k$, i will create a cryptographic onion by encrypting m with the public key corresponding to P_k to get o_k, then encrypting o_k with the public key corresponding to P_{k-1}, etc. When the onion is transmitted, each party will use its private key to decrypt the onion, "peeling" away the outermost layer. When the onion reaches the recipient P_j, only the inner layer will remain. This structure guarantees that only the final intended recipient can decrypt the original message.

In this thesis, we assume the synchronous model of communication. Each onion is received instantaneously once sent, meaning at a given round r, each party P_i will have already received all onions sent to it in round $r-1$.

By an *onion routing protocol*, we mean a protocol in which the parties use an onion encryption scheme to form and process all message packets. We also consider the protocol operating over some set of possible inputs Σ, as described below. We denote such a protocol as $\Pi(1^\lambda, \Sigma)$.

Adversary Model. We define the three standard adversary models: the network adversary, the passive adversary, and the active adversary. All adversaries can see all network traffic during a run of the protocol; in fact, the *network adversary* can observe exactly that and no more.

A *passive adversary* can also corrupt a fraction of the parties. When we say $\mathcal{A}$ corrupts a party, we mean $\mathcal{A}$ can see all actions that party performs but cannot control the party to deviate from the protocol.

An *active adversary* can control a fraction of the parties. By this we mean $\mathcal{A}$ can control those parties freely, including deviating from the protocol.

We are primarily concerned with the passive adversary corrupting some fraction $\frac{1}{f(\lambda)}$ of the parties. Note that in this setting, we do not assume the existence of a trusted third party, since the adversary can corrupt any of the parties with some probability. Furthermore, since the passive adversary cannot alter the behavior of the corrupted parties, they behave identically to the honest parties, and the protocol cannot determine which parties are corrupted.

Inputs. We define the *message space* $\mathcal{M}$ as the set of possible messages the participants send. The length of messages in $\mathcal{M}$ is polynomially bounded by λ, and we assume all messages are the same length.

Each party is represented by a unique integer value, and the set of all parties is represented by $[N]$. An *input* to a party is a set of message/recipient pairs $\{(m, i)\}$ where each $m \in \mathcal{M}$ and $i \in [N]$. Each pair indicates a message and the party to

whom to send it.

A *vector of inputs* to the protocol is a vector $\sigma = \{v_1, ..., v_N\}$ where each v_i is an input to party i.

We denote the set of all possible inputs to a protocol as Σ.

In this paper, we consider the SimpleIO setting, formally introduced in [ALU18]. By this, we mean that the set of inputs is constrained to those where each user is instructed to send one message and each user expects to receive exactly one message. This setting, or something even more stringent, is used by most provably secure protocols [vdHLZZ15, TGL+17, KCDF17].

Input equivalence. We say two inputs σ_1, σ_2 are equivalent with respect to an adversary $\mathcal{A}$ if:

1. For each corrupted party P_i, the input to P_i in σ_1 is identical to the input to P_i in σ_2.

2. Additionally, for each corrupted party P_i, the set of onions received by P_i in σ_1 is identical to the set of onions received by P_i in σ_2.

We introduce this notion of equivalence to define a reasonable notion of anonymity. Note that the adversary can trivially distinguish between non-equivalent inputs, simply by examining the information learned from corrupted parties. Therefore, any protocol can only hope to protect equivalent inputs, and it is only these inputs we consider.

Dummy onion. We say that an onion is a *dummy onion* if it is formed by an honest party, but it is *not* generated by running FormOnion on an input message. Intuitively, dummy onions do not contain meaningful content and serve to muddle the network traffic. We sometimes refer to onions created by running FormOnion on an input message as *genuine*.

Views. We say the adversary $\mathcal{A}$'s *view* associated with a run of the protocol Π

on input σ is all information $\mathcal{A}$ sees associated with that run of protocol, denoted by $V^{\Pi,\mathcal{A}}(\sigma)$. For the passive adversary, this information includes traffic over all network links, in addition to the states and computation of the corrupted servers. More specifically, for each onion routed through a corrupted server, $\mathcal{A}$ can see where the onion previously came from and where it is sent to next. Additionally, for any onion received by a corrupted server, the adversary can determine whether the message contained in it is a dummy message or a genuine message. However, without additional information, the adversary cannot discover that message's origin.

2.2 Definitions (anonymity)

These definitions use the notions of input equivalence and views, defined in 2.1.

Definition 1. Statistical anonymity

We say an onion routing protocol $\Pi(1^\lambda, \Sigma)$ is *statistically anonymous* from an adversary $\mathcal{A}$ corrupting a fraction $0 \leq \kappa < 1$ of the parties if, for any equivalent inputs $\sigma_1, \sigma_2 \in \Sigma$, $V^{\Pi,\mathcal{A}}(\sigma_1)$ is statistically indistinguishable from $V^{\Pi,\mathcal{A}}(\sigma_2)$.

We usually operate under a game-based anonymity definition, in which the adversary is computationally bounded. For a precise definition, see [ALU20]. A more brief description is provided here.

Definition 2. Game-based anonymity

Consider the following game consisting of an onion routing protocol Π and an adversary $\mathcal{A}$ with polynomially bounded computing power. $\mathcal{A}$ picks any two inputs σ_1, σ_2 that are equivalent with respect to $\mathcal{A}$. One of the two inputs is chosen uniformly at random; let this be σ_i. The protocol Π runs on this input σ_i, and $\mathcal{A}$ is able to see the view of the protocol, $V^{\Pi,\mathcal{A}}(\sigma_i)$. The adversary then guesses whether σ_i is equal to σ_1 or σ_2. If the adversary guesses correctly, it wins.

An onion routing protocol Π is anonymous from an adversary $\mathcal{A}$ if and only if $\mathcal{A}$ wins this game with probability at most $\frac{1}{2} + \frac{1}{f(\lambda)}$, where λ is the security parameter and $f(\lambda)$ grows faster than any polynomial in λ.

2.3 Definitions (efficiency measures)

Definition 3. Onion cost

We say the *onion cost* associated with a protocol is the expected value of the total number of onion transmissions across all servers. We denote the onion cost of a protocol Π with adversary $\mathcal{A}$ as $\mathsf{OC}^{\Pi,\mathcal{A}}(\Sigma)$, where Σ is the set of all possible inputs. The expectation is taken over the inputs in the input set, which are chosen uniformly at random, and the randomness of the protocol.

Definition 4. Server load

The *server load* of a protocol given an input σ and adversary $\mathcal{A}$ is the maximum number of onions transmitted by a server in any round.

Definition 5. Round complexity

The *round complexity* of a protocol Π with input set Σ and adversary $\mathcal{A}$ is the average over all possible inputs of the number of rounds from the first onion being transmitted to the last onion being received.

Chapter 3

Results

3.1 Round Complexity

Consider an onion routing protocol over N parties with security parameter λ operating in r rounds, with a passive adversary $\mathcal{A}$ corrupting a fraction $\kappa = \frac{1}{f(\lambda)}$ of the parties for some function f.

We consider only functions $f(\lambda)$ that are generally increasing, since security is defined with respect to adversaries who corrupt a constant fraction of the parties.

We wish to prove a lower bound on the number of rounds required for anonymity. Recall that a protocol is not anonymous if the adversary can with non-negligible advantage distinguish between any two inputs σ_0 and σ_1, whose inputs are identical for corrupted parties. Note: the adversary can trivially distinguish between inputs that differ for corrupted parties, so we cannot hope to achieve anonymity in these cases and do not consider them.

Theorem 1. If an onion routing protocol with security parameter λ operating in r rounds is anonymous from the passive adversary corrupting a fraction $\frac{1}{f(\lambda)}$ of the parties, and for any input, every honest onion corresponding to a message in that input has a non-negligible probability of being formed in the first round, r must be $\omega\left(\frac{\log \lambda}{\log f(\lambda)}\right)$.

Proof. Let σ_0 be an input under which party i sends a message to party j and σ_1 be an

input in which i *does not* send a message to j. If $\mathcal{A}$ can trace a non-dummy message received by j back to its original sender i, $\mathcal{A}$ knows that the input must have been σ_0. Let o be a genuine message-bearing onion originating at i and received by j in σ_0. If o was created at round 1, and $\mathcal{A}$ can trace back its path to its honest sender, $\mathcal{A}$ will know that that sender must have created o, since there were no previous rounds. If $\mathcal{A}$ corrupts j, $\mathcal{A}$ can tell whether any message j receives is a dummy or genuine. Therefore, if $\mathcal{A}$ corrupts j, $\mathcal{A}$ can determine which of the onions j receives is o. If $\mathcal{A}$ corrupts all the users that o passes through, $\mathcal{A}$ can determine that i sent o to j, and $\mathcal{A}$ knows the input must have been σ_0. Therefore, if these events occur, the adversary can distinguish between σ_0 and σ_1 with non-negligible advantage, and the protocol is not anonymous. More precisely:

Let E_1 be the event that o is created in round 1 of r rounds.

Let E_2 be the event that $\mathcal{A}$ corrupts all users o passes through.

Let E_3 be the event that $\mathcal{A}$ corrupts j.

If all three of E_1, E_2, E_3 occur, $\mathcal{A}$ wins. Therefore, if $Pr[E_1, E_2, E_3]$ is non-negligible, the protocol is not anonymous. We give a bound on this probability below.

Since the adversary cannot alter the behavior of the corrupted parties, the corrupted parties look identical to the honest parties, and the protocol can do no better than choosing a routing path uniformly at random. The protocol minimizes the probability of choosing a routing path consisting entirely of corrupted parties by choosing a routing path *without replacement*. Note that the number of parties o passes through is at most r, since the protocol operates in only r rounds. Therefore,

$$Pr[E_2|E_1] \leq \frac{\binom{\#\text{ corrupt users}}{r}}{\binom{\#\text{ total users}}{r}} = \frac{\binom{\frac{N}{f(\lambda)}}{r}}{\binom{N}{r}}$$

And, using the fact that $\frac{n^k}{k^k} \leq \binom{n}{k} < \frac{(en)^k}{k^k}$, we have:

$$\frac{\binom{\frac{N}{f(\lambda)}}{r}}{\binom{N}{r}} > \frac{\left(\frac{N}{f(\lambda)}\right)^r}{(eN)^r} = \left(\frac{1}{ef(\lambda)}\right)^r$$

Recall that E_3 is the event that $\mathcal{A}$ corrupts j, which happens with probability $\frac{1}{f(\lambda)}$. Therefore:

$$\begin{aligned} Pr[E_1, E_2, E_3] &= Pr[E_3|E_1, E_2] \cdot Pr[E_2|E_1] \cdot Pr[E_1] \\ &= Pr[E_3] \cdot Pr[E_2|E_1] \cdot Pr[E_1] \\ &= \frac{1}{f(\lambda)} \cdot \left(\frac{1}{ef(\lambda)}\right)^r \cdot Pr[E_1] \end{aligned}$$

$Pr[E_1]$, the probability that onion o is created in the first round, is non-negligible by assumption. Therefore, $Pr[E_1, E_2, E_3]$ is non-negligible if and only if $\left(\frac{1}{ef(\lambda)}\right)^r$ is non-negligible (for now, we assume $\frac{1}{f(\lambda)}$ is non-negligible; otherwise, the adversary should be equivalent to the network adversary).

Let $r = \frac{\log \lambda}{\log f(\lambda)} = \log_{f(\lambda)} \lambda$ by change of base formula. Plugging in, we have:

$$\left(\frac{1}{ef(\lambda)}\right)^r = \frac{1}{e^{\frac{\log \lambda}{\log f(\lambda)}}} \cdot \frac{1}{f(\lambda)^{\log_{f(\lambda)} \lambda}}$$

$$= \frac{1}{\lambda^{\frac{1}{\log f(\lambda)}} \cdot \lambda}$$

Notice that since $f(\lambda)$ is increasing, there is some n such that for all $\lambda > n$, $\lambda^{\frac{1}{\log f(\lambda)}} < \lambda$. Therefore,

$$Pr[E_2|E_1] \geq \frac{1}{\lambda^{\frac{1}{\log f(\lambda)}} \cdot \lambda} = \Omega\left(\frac{1}{\lambda^2}\right)$$

And since $Pr[E_1]$ and $Pr[E_3]$ are also non-negligible, the adversary can distinguish between inputs with non-negligible probability, and the protocol is not anonymous. Therefore, $\omega\left(\frac{\log \lambda}{\log f(\lambda)}\right)$ rounds are insufficient for anonymity from the passive adversary corrupting a fraction $\frac{1}{f(\lambda)}$ of the parties. □

Corollary 1. The same bound applies for a passive adversary that can observe network traffic over only a constant fraction of links.

If all servers in a message's routing path are corrupted, the adversary can determine the message's origin without needing to see any of the network traffic. Therefore,

the above proof holds even if the adversary can see only a constant fraction, or even none of the links.

3.2 Round complexity with server load constraint

Theorem 1 tells us that even if we choose the same routing path for all messages, shuttling them through the same servers in sequence (like in example 1.2.3), we still need $\omega\left(\frac{\log \lambda}{\log f(\lambda)}\right)$ rounds for anonymity. However, this approach is very inefficient, requiring a server load of N. It is then natural to ask what the round complexity must be if the server load is constrained in some way. In this section, we prove a lower bound on the round complexity required for anonymity for an onion routing protocol with a server load of at most α. In other words, at most α onions can meet at a given server at a time. Here, we consider protocols choosing routing paths uniformly at random with replacement and the passive adversary corrupting a constant fraction of the parties.

We first introduce a lemma for anonymity against the less powerful network adversary, which can view all network traffic but cannot corrupt any parties.

Lemma 1. If an onion routing protocol with maximum server load α is anonymous against a network adversary in the $\mathsf{SimpleIO}$ setting, every honest onion must be routed through at least $\Omega\left(\frac{\log N}{\log \alpha}\right)$ servers.

Proof. Recall that in the $\mathsf{SimpleIO}$ setting, each of the N parties sends and receives exactly one message. In order for a protocol to be anonymous against a network adversary observing all network traffic, each honest onion o must have a potential path formed by the network traffic linking it back to every honest party, of which there are $(1-\kappa)N$. In other words, there must be some way to trace o back to every honest party in a way that is consistent with the network traffic. Otherwise, the adversary can say with certainty that o did not originate at some honest party, and the protocol is not anonymous.

More formally, we define a *network path* $P^r = (n_0, n_1, ..., n_r)$ as a tuple of servers. P^r is a valid network path for a run of protocol Π if there is some k such that for every

$0 \leq i \leq r-1$, server n_i transmits an onion to server n_{i+1} in round $k+i$. Intuitively, if $(n_0, n_1, ..., n_r)$ is a valid network path, then an onion at server n_r at round $k+r$ could have originated at server n_0 at round k.

Let o be an honest onion. We define $S^i(o)$ as the set of servers from which it appears o could have originated, given the network traffic up to round i. More formally,

$$S^i(o) = \{S_j \mid \exists\ P^k = (n_0, ..., n_k) \text{ s.t. } n_k = S\}$$

where S is the server to which o is transmitted in round i, and P^i is a valid network path.

We show by induction on i that $|S^i(o)| \leq \sum_{j=0}^{i} \alpha^j$, where α is the server load. In the first round, onion o is transmitted from its origin to an intermediate server. At this intermediate server, o meets at most $\alpha - 1$ other onions. There are then at most $\alpha - 1$ network paths from those onions' origins to the intermediate server, along with the path o itself followed. There is also the trivial path from the intermediate server to itself. Therefore, $|S^1(o)| \leq \alpha - 1 + 1 + 1 = \alpha^1 + 1$.

Assume as our inductive hypothesis that for a fixed and arbitrary $k \geq 1$, $|S^k(o)| \leq \sum_{j=0}^{k} \alpha^j$.

Consider the server S where o is at the end of round $k+1$. At this server, o meets at most $\alpha - 1$ other onions. For any of these other onions o', $|S^k(o')| \leq \sum_{j=0}^{k} \alpha^j$ by assumption. For o itself, $|S^k(o)| \leq \sum_{j=0}^{k} \alpha^j$ by assumption. There are at most as many valid network paths to S ending in round i as there are valid network paths to servers transmitting onions to S in round i, plus the one trivial path from S to itself. Therefore, $|S^{k+1}(o)| \leq 1 + \alpha \sum_{j=0}^{k} \alpha^j = \sum_{j=0}^{k+1} \alpha^j$.

We've thus shown that for all $i \geq 1$, $|S^i(o)| \leq \sum_{j=0}^{i} \alpha^j$.

In order to be anonymous against the network adversary, there must be a network path from the destination of o to every server. Therefore, o must be routed through at least r servers, where $\sum_{j=0}^{r} \alpha^j \geq N$. Solving for

$$\sum_{j=0}^{r} \alpha^j \leq (r+1)\alpha^r = N$$

We have:

$$r \log \alpha + \log(r+1) = \log N$$

$$r = \Omega\left(\frac{\log N}{\log \alpha}\right)$$

□

Corollary 2. If an onion routing protocol with maximum server load α is anonymous against a passive adversary corrupting a fraction κ of the servers in the SimpleIO setting, every honest onion must be routed through at least $\Omega\left(\frac{\log((1-\kappa)N)}{\log \alpha}\right)$ *honest* servers.

Proof. We adapt Lemma 1 for our use here. The passive adversary can view all network traffic, along with all actions corrupted servers perform. Therefore, if an honest onion o passes through a corrupted server at round $i+1$, the number of possible origins of o does not increase and $|S^i(o)| = |S^{i+1}(o)|$.

Since the adversary knows the inputs to corrupted parties, it is not distinguishing if it can trace an onion back to a corrupted origin. Therefore, the set of onions from which o could have originated need only contain all honest servers, of which there are $(1-\kappa)N$.

Therefore, each honest onion must pass through at least $\Omega\left(\frac{\log((1-\kappa)N)}{\log \alpha}\right)$ *honest* servers in order for the protocol to be anonymous. □

We are now ready to prove the main result of this section.

Theorem 2. Let Π be an onion routing protocol with N parties and maximum server load α operating in the SimpleIO setting, where the servers in each onion's routing path are chosen uniformly at random with replacement. If Π is anonymous against the passive adversary corrupting a *constant* fraction $\kappa = \frac{1}{f(\lambda)}$ of the servers, the round complexity of Π is

$$\omega\left(\frac{\log \lambda + \log h}{\log f(\lambda)} + h\right)$$

where $h = \frac{\log((1-\kappa)N)}{\log \alpha}$.

Proof. We show that if Π operates in $L = c(\frac{\log\lambda + \log h}{\log f(\lambda)} + h)$ rounds, where c is some constant, some honest onion passes through fewer than h honest servers with non-negligible probability. By Corollary 2, with overwhelming probability, each honest onion must pass through at least h honest servers in order for Π to be anonymous.

Let o be a message bearing onion formed by an honest party. Let $\mathcal{E}$ denote the event that o is routed through fewer than h honest servers, where the servers in the routing path are chosen uniformly at random with replacement. Summing over the probability of o being routed through exactly i honest servers, for $i < h$, we have:

$$Pr(\mathcal{E}) = \sum_{i=0}^{h-1} \binom{L}{i} \kappa^{L-i}(1-\kappa)^i$$

$$\geq h\kappa^{L-i}(1-\kappa)^i$$

Either κ or $1-\kappa$ is less than or equal to $\frac{1}{2}$; let this smaller value be $\frac{1}{k}$. We then have

$$h\kappa^{L-i}(1-\kappa)^i \geq \frac{h}{k^L}$$

And plugging in L, we have

$$Pr(\mathcal{E}) \geq \frac{h}{k^L} = \frac{h}{k^{c\left(\frac{\log\lambda + \log h}{\log f(\lambda)} + h\right)}}$$

$$= \frac{h}{\lambda^{\frac{c\log k}{\log f(\lambda)}} h^{\frac{c\log k}{\log f(\lambda)}} e^{ch\log k}}$$

And, recalling that $h = \frac{\log((1-\kappa)N)}{\log\alpha}$, and that $c, f(\lambda)$, and k are constants, for some c_1, c_2 this is equal to

$$\frac{h}{\lambda^{c_1} h^{c_1} e^{hc_2}} = \frac{h}{\lambda^{c_1}\left(\frac{\log((1-\kappa)N)}{\log\alpha}\right)^{c_1}((1-\kappa)N)^{\frac{c_2}{\log\alpha}}}$$

$$\geq \frac{h}{\lambda^{c_1}(\log((1-\kappa)N))^{c_1}((1-\kappa)N)^{c_2}}$$

$$= \frac{\log((1-\kappa)N)}{poly(\lambda)\log\alpha}$$

$$\geq \frac{1}{poly(\lambda)}$$

for some polynomial $poly(\lambda)$. Note that for our purposes, we can assume the server load α is $O(2^\lambda)$, since if all packets can meet at a single honest server at the same time, anonymity is easily achievable as illustrated in the motivating examples.

Therefore, if Π operates in $L = c(\frac{\log\lambda+\log h}{\log f(\lambda)} + h)$ rounds, onion o passes through fewer than h honest servers with non-negligible probability. A passive adversary can then distinguish between inputs and Π is not anonymous. Thus, Π must operate in $\omega(\frac{\log\lambda+\log h}{\log f(\lambda)} + h)$ rounds. □

Remark: In the proof of Theorem 2, we consider the servers in the routing path as chosen uniformly at random *with replacement*, whereas in Theorem 1, we consider them *without replacement*. This is because in Theorem 1, the protocol minimizes the probability of the adversary corrupting all servers in a routing path by avoiding repeating servers, since this maximizes the probability of passing through at least one honest server. In the proof of Theorem 2, the protocol wants each onion to pass through at least h honest servers, instead of just 1 honest server. It then seems best to choose the routing path with replacement, since otherwise each time we add an honest server to the routing path, the probability of later choosing an honest server decreases.

3.2.1 Tightness

We suspect that the bound in Theorem 3 is asymptotically tight for event $\mathcal{E}$ as defined above, though there might be another more likely event leading to the adversary distinguishing between inputs.

Observe that if $p(L)$ is a function of L representing an overestimate of $Pr(\mathcal{E})$, where $p(L) \geq Pr(\mathcal{E})$ for all L, then if $p(L)$ is negligible, $Pr(\mathcal{E})$ must be negligible as well. We show that for such a $p(L)$, our L^* as given in Theorem 3 is the asymptotically

greatest L such that $p(L)$ is non-negligible. Recall that

$$Pr(\mathcal{E}) = \sum_{i=0}^{h-1} \binom{L}{i} \kappa^{L-i}(1-\kappa)^i$$

Since $\binom{L}{i} \leq L^i$, and $1 - \kappa \leq 1$,

$$\sum_{i=0}^{h-1} \binom{L}{i} \kappa^{L-i}(1-\kappa)^i \leq hL^h\kappa^{L-h}$$

And this will be our overestimate $p(L) = hL^h\kappa^{L-h}$. For some constant c, we need $p(L) \geq \frac{1}{\lambda^c}$. We can plug in and solve for L:

$$hL^h\kappa^{L-h} \geq \frac{1}{\lambda^c}$$

$$\log h + h \log L + (L-h) \log \kappa \geq -c \log \lambda$$

$$L \log f(\lambda) - h \log L \leq \log h + c \log \lambda + h \log f(\lambda)$$

And the $h \log L$ term becomes asymptotically small, so we have

$$L \leq \Theta\left(\frac{c \log \lambda + \log h}{\log f(\lambda)} + h\right)$$

Since $p(L) \geq Pr(\mathcal{E})$ for all L, and $L = \Theta\left(\frac{c \log \lambda + \log h}{\log f(\lambda)} + h\right)$ is the asymptotically greatest value of L such that $p(L)$ is non-negligible, L^* is tight for $Pr(\mathcal{E})$.

Chapter 4

Conclusion

4.1 Implications

Our results provide bounds on the round complexity required for anonymity against the passive adversary. Notably, Theorem 1 shows that the protocol Π_p is tight in terms of round complexity [ALU18].

While we assume the SimpleIO setting, the attack we consider in the proof of Theorem 1 involves only tracing a given onion back to its sender. This attack is not unique to the SimpleIO setting, and with a bit more work this result should generalize to more settings. In the proof of Theorem 2, we want a given onion to "look like" it could have originated at any of the honest servers. We can likely adapt this to other input settings, where instead we just need the onion to look like it could have originated from some subset of the honest servers, where the subset depends on the setting.

Our bounds apply not just to anonymity but also to differential privacy [DR14]. Informally, a protocol is differentially private if, for any neighboring inputs, the distributions of the adversary's view given each of the inputs vary by a multiplicative factor ϵ and an additive factor δ, where δ is negligible in the security parameter. We say two inputs are neighboring if the input to each party is the same except for one message-recipient pair. For example, in the first input Alice sends to Bob, and in the second input, Alice sends to Carol. In our proofs, we show that an adversary can

distinguish between inputs that differ by only one message-recipient pair. Therefore, our bounds hold for differential privacy as well.

4.2 Future Work

First, we would like to generalize Theorem 2 to the adversary corrupting a non-constant fraction of the servers and ideally tighten the bound as well. Though it seems to be tight for the anonymity breaking event we consider, it is possible that we could find a different anonymity breaking event that occurs with higher probability.

As noted earlier, we suspect that the protocol can do no better than choosing routing paths uniformly at random *without* replacement when avoiding the attack detailed in the proof of Theorem 1, and no better than choosing routing paths uniformly at random *with* replacement when avoiding the attack detailed in the proof of Theorem 2. We hope to formalize this in the future and provide a proof of this optimality, rather than relying on assumptions.

Additionally, we use the SimpleIO setting throughout this thesis, where each party sends and receives exactly one message. We hope to extend our results more generally.

Bibliography

[ALU18] Megumi Ando, Anna Lysyanskaya, and Eli Upfal. Practical and provably secure onion routing. In Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella, editors, *ICALP 2018*, volume 107 of *LIPIcs*, pages 144:1–144:14. Schloss Dagstuhl, July 2018.

[ALU20] Megumi Ando, Anna Lysyanskaya, and Eli Upfal. On the complexity of anonymous communication through public networks. Manuscript, in submission, 2020.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145. IEEE Computer Society Press, October 2001.

[Cha81] David L Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–90, 1981.

[CL05] Jan Camenisch and Anna Lysyanskaya. A formal treatment of onion routing. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 169–187. Springer, Heidelberg, August 2005.

[DMMK18] Debajyoti Das, Sebastian Meiser, Esfandiar Mohammadi, and Aniket Kate. Anonymity trilemma: Strong anonymity, low bandwidth overhead, low latency - choose two. In *2018 IEEE Symposium on Security and Privacy*, pages 108–126. IEEE Computer Society Press, May 2018.

[DMS04] Roger Dingledine, Nick Mathewson, and Paul F. Syverson. Tor: the second-generation onion router. In *Proceedings of the 13th USENIX Security Symposium, August 9-13, 2004, San Diego, CA, USA*, pages 303–320, 2004.

[DR14] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014.

[Gol98] Oded Goldreich. Secure multi-party computation. *Manuscript. Preliminary version*, 78, 1998.

[JWJ+13] Aaron Johnson, Chris Wacek, Rob Jansen, Micah Sherr, and Paul F. Syverson. Users get routed: traffic correlation on tor by realistic adversaries. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM CCS 13*, pages 337–348. ACM Press, November 2013.

[KCDF17] Albert Kwon, Henry Corrigan-Gibbs, Srinivas Devadas, and Bryan Ford. Atom: horizontally scaling strong anonymity. In *Proceedings of the 26th Symposium on Operating Systems Principles, Shanghai, China, October 28-31, 2017*, pages 406–422, 2017.

[SEF+17] Yixin Sun, Anne Edmundson, Nick Feamster, Mung Chiang, and Prateek Mittal. Counter-RAPTOR: Safeguarding tor against active routing attacks. In *2017 IEEE Symposium on Security and Privacy*, pages 977–992. IEEE Computer Society Press, May 2017.

[TGL+17] Nirvan Tyagi, Yossi Gilad, Derek Leung, Matei Zaharia, and Nickolai Zeldovich. Stadium: a distributed metadata-private messaging system. In *Proceedings of the 26th Symposium on Operating Systems Principles, Shanghai, China, October 28-31, 2017*, pages 423–440, 2017.

[vdHLZZ15] 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, Monterey, CA, USA, October 4-7, 2015*, pages 137–152, 2015.

[WSJ+18] Ryan Wails, Yixin Sun, Aaron Johnson, Mung Chiang, and Prateek Mittal. Tempest: Temporal dynamics in anonymity systems. *PoPETs*, 2018(3):22–42, 2018.