FUNDAMENTAL CONSTRAINTS AND PROVABLY SECURE CONSTRUCTIONS OF ANONYMOUS COMMUNICATION PROTOCOLS

by

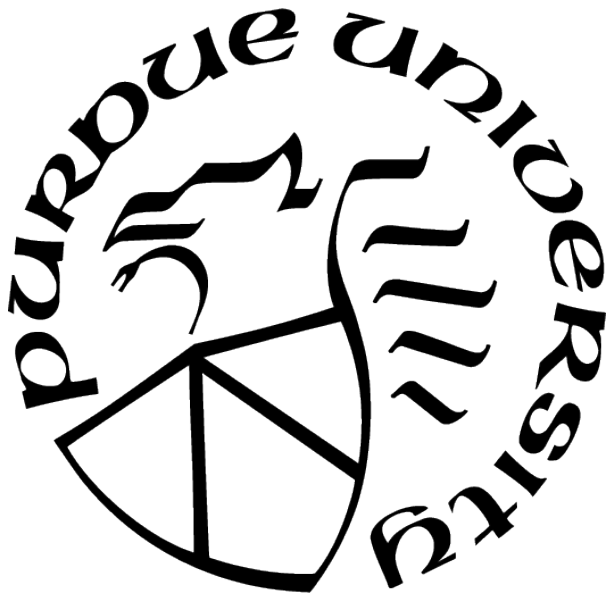
Debajyoti Das

A Dissertation

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the degree of

Doctor of Philosophy



Department of Computer Science

West Lafayette, Indiana

August 2021

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

Dr. Aniket Kate, Chair

School of Computer Science

Dr. Mikhail Atallah

School of Computer Science

Dr. Jeremiah Blocki

School of Computer Science

Dr. Wojciech Szpankowski

School of Computer Science

Approved by:

Dr. Kihong Park

Remembering my dad who asked for so little but gave so very much...

ACKNOWLEDGMENTS

Many people have influenced me during my time as a doctoral student, and it is definitely not an easy task to acknowledge their impact adequately in a few words.

I want start by thanking my advisor Aniket Kate for his guidance, encouragement, and support in the past six years. He always gave me sufficient freedom to explore different areas of research and choose my own research agenda. Many people claim that after several years doctoral students start looking like their advisors; I wish I could carry some of his research skills with me in my upcoming career as a researcher.

I also want to thank all my colleagues with whom I have worked with on various projects, especially Sebastian Meiser and Esfandiar Mohammadi; I will cherish the discussions, the debates, and most importantly the Starcraft games that I played with them. Not only that, they positively influenced my scientific thinking and research methodologies through their constructive criticisms and skepticisms.

I want to thank all my labmates from Freedom Research Lab — Aditya Bhat, Sze Yiu Chau, Tiantian Gong, Mahimna Kelkar, Duc V. Le, Donghang Lu, Easwar V. Mangipudi, Mohsen Minaei, and Pedro Moreno-Sanchez. I will really miss the philosophical discussions with them.

I would like to thank Navneet Joshi, Sourav Sekhar Bera, Shouvik Bhattacharya, Jenan Almulla, Debipriya Das, Somosmita Mitra, Shaurya Sharma, Imon Banerjee, Sujay Somaiah, and Basavesh Sivakumar for always supporting me and helping me through many difficult times of my life.

I want to thank my family for their never-ending love and support. Without them I would not be the person that I am today.

Last but not the least, I want to thank all the living beings in the universe who endorse rational thinking and logical arguments...

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Anonymous communication networks (ACNs) are critical to communication privacy over the internet as they enable individuals to maintain their privacy from untrusted intermediaries and endpoints. Typically, ACNs involve messages traveling through some intermediaries before arriving at their destinations, and therefore they introduce network latency and bandwidth overheads.

The goal of this work is to investigate the fundamental constraints of anonymous communication (AC) protocols. We analyze the relationship between bandwidth overhead, latency overhead, and sender anonymity or recipient anonymity against a global passive (network-level) adversary. We confirm the widely believed trilemma that an AC protocol can only achieve two out of the following three properties: strong anonymity (i.e., anonymity up to a negligible chance), low bandwidth overhead, and low latency overhead.

We further study anonymity against a stronger global passive adversary that can additionally passively compromise some of the AC protocol nodes. For a given number of compromised nodes, we derive as a necessary constraint a relationship between bandwidth and latency overhead whose violation make it impossible for an AC protocol to achieve strong anonymity. We analyze prominent AC protocols from the literature and depict to which extent those satisfy our necessary constraints. Our fundamental necessary constraints offer a guideline not only for improving existing AC systems but also for designing novel AC protocols with non-traditional bandwidth and latency overhead choices.

Using the guidelines indicated by our fundamental necessary constraints we provide two efficient protocol constructions. First, we design a mixnet-based AC protocol *Streams* that provides provable mixing guarantees with the expense of latency overhead. Streams realizes a trusted third party stop-and-go mix as long as each message stays in the system for $\omega(\log \eta)$ rounds. Second, we offer a DC-net based design *OrgAn* that can provide strong sender anonymity with constant latency at the expense of bandwidth overhead. OrgAn solves the problem of regular requirements of key and slot agreement present in typical DC-net based protocols, by utilizing a client/relay/server architecture.

1. INTRODUCTION

Anonymous communication networks (ACNs) [1–15] are critical to communication privacy over the Internet as they enable individuals to maintain their privacy from untrusted intermediaries and endpoints. Typically, ACNs involve messages traveling through some intermediaries before arriving at their destinations, and therefore they introduce network latency and bandwidth overheads. Even after almost four decades of work, the search for an optimal overhead ACN design is still unfinished: Anonymity requires the company of others interested in anonymity, which indeed makes it a hard problem to solve.

In the present day, millions of users from all over the world employ anonymous communication networks, such as Tor [16], to protect their privacy over the Internet. The design choice made by the Tor network to keep the latency and bandwidth overheads small has made it highly attractive to its geographically diverse user-base. However, over the last decade, the academic literature [17–25] has demonstrated Tor's vulnerability to a variety of traffic correlation attacks. In fact, Tor also has been successfully attacked in practice [26]. It is widely accepted that low-latency low-bandwidth overhead of anonymous communication (AC) protocols, such as Tor [13], can only provide a weak form of anonymity [27].

In the anonymity literature, several AC protocols were able to overcome this security barrier to provide a stronger anonymity guarantee (cryptographic indistinguishability-based anonymity [28, 29]) by either increasing the latency overhead or the bandwidth overhead. For example, high-latency approaches (such as mixnet-based systems [2, 3, 30–34]) can ensure strong anonymity by introducing significant communication delays for users messages. Protocols such as dining Cryptographers network [12] and its extensions [5, 11, 15, 35, 36]) are academically interesting because they can provide robustness against compromisation *using some pre-established coordination among users*. However, even with such *user coordination* techniques, they can provide strong anonymity by adding copious amount of noise (or dummy) messages.

There have been a few efforts to propose approaches [1, 2, 5, 8, 37, 38] that try to provide anonymity by simultaneously introducing latency and bandwidth overhead. However, it is

not clear how to balance such system parameters to ensure strong anonymity while preserving practical performance.

In general, in the last 35 years a significant amount of research efforts have been put towards constructing novel AC protocols, deploying them, and attacking real-world AC networks. However, unlike other security fields such as cryptography, our understanding regarding the fundamental limits and requirements of AC protocols remains limited. This work takes some important steps towards answering fundamental question associated with anonymous communication. "Can we prove that strong anonymity cannot be achieved without introducing large latency or bandwidth overhead? When we wish to introduce the latency and bandwidth overheads simultaneously, do we know the overhead range values that still fall short at providing stronger anonymity?"

This dissertation answers the above questions by deriving necessary constraints in terms of latency and bandwidth overhead that are absolutely necessary to achieve strong sender or recipient anonymity against global network level adversaries. Further, it evaluates the trade-off between latency and bandwidth overhead to achieve strong anonymity. In short, this thesis focuses on demonstrating the following statement:

> *There exists a fundamental necessary constraint (in terms of latency and bandwidth overhead) for anonymous communication protocols to achieve strong anonymity against global network level adversaries.*

1.1 Contributions

As a main contribution, this work confirms a previously conjectured [1, 7] relationship between bandwidth overhead, latency overhead and anonymity. We find that there are fundamental bounds on sender and recipient anonymity properties [28, 29, 39] of a protocol that directly depend on the introduced bandwidth and latency overheads.

In this work, we present a generic model of AC protocols using Petri nets [40, 41] such that different instantiations of this model will represent different AC protocols, covering most practical AC systems in the literature. We derive *upper* bounds on anonymity as functions of bandwidth overhead and latency overhead, against two prominent adversary classes:

global passive network-level adversaries and strictly stronger adversaries that additionally (passively) compromise some protocol parties (e.g., relays in case of Tor). These bounds constitute necessary constraints for anonymity. Naturally, the constraints are valid against any stronger adversary class as well.

For both adversary classes, we analyze two different user distributions (i.e., distributions that determine at which time or rate users of the AC protocol send messages): (i) synchronized user distributions, where users globally synchronize their messages, and (ii) unsynchronized user distributions, where each user locally decides when to send his messages independent of other users.

We further identify a general class of techniques from the line of works [4–6,9,10] that can provide better anonymity than classical mix-nets, which we call user coordination: *assuming some form of (free) coordination among a set of N users, $h+1 \leq N$ users send a packet each for some single (actual) message such that (i) the receiver can retrieve the actual message only after receiving all the $h+1$ packets, and that (ii) the receiver of the message cannot distinguish who among the $h+1$ users actually sent the message.* The goal here is not to keep user coordination practical; rather, we define the notion in this way to capture all efficient instantiations of similar techniques. One prominent example is given by DC-nets. DC-nets use shared keys/coins to produce dummy messages (corresponding to our shares) that allow the receiver to reconstruct the actual message.

In this work, we differentiate between protocols with user coordination and protocols without in our analysis. We show that even protocols with user coordination must either use an excessive bandwidth overhead (every user sends a share for every real message by any other user) or adhere to our anonymity trilemma. We provide a comparative analysis between the two class of protocols.

Formal lower bounds. We first analyze the trade-off between latency overhead and bandwidth overhead required to achieve *strong anonymity*, i.e., anonymity up to a negligible (in a security parameter η) chance of failure. For any AC protocol where only a fraction of $\beta \in [0,1]$ users send noise messages per communication round, and where messages can only remain in the network for $\ell \geq 0$ communication rounds, we find that against a global

network-level adversary no protocol can achieve strong anonymity if $2\beta\ell < 1 - 1/\text{poly}(\eta)$ even when all the protocol parties are honest. For $\mathsf{c} > 0$, we show that strong anonymity is impossible for constant latency ($\ell \in \Theta(1)$).

In the case where a strictly stronger adversary additionally passively compromises c (out of K) protocol parties, we show that strong anonymity is impossible if $2(\ell - \mathsf{c})\beta < 1 - 1/\text{poly}(\eta)$ (for $\mathsf{c} < \ell$), or $2\beta\ell < 1 - 1/\text{poly}(\eta)$ and $\ell \in \mathcal{O}(1)$ (for $\mathsf{c} \geq \ell$), when protocols do not use user coordination.

Even when protocols use user coordination, if half of all K nodes are compromised ($\mathsf{c} = \mathsf{K}/2$), ACNs with strong anonymity cannot have a latency that is logarithmic ($\ell \leq \log(\mathsf{K})$) in the number K of protocol nodes. For the Anytrust setting [5] where all but a constant amount of protocol nodes are compromised, strong anonymity requires a minimal latency in the order of $\sqrt{\mathsf{K}}$.

Proof formalism. Our proof technique provide several key insights that might be of independent interest: novel necessary constraints for anonymity as well as novel design goals for an ideal AC protocol. Moreover, we provide intuitive readings of our formal theorems that summarize their key insights. The proof technique used in this work can be used to derive bounds against other adversary classes (weaker than global passive adversaries) as well.

Lessons learned. We show the correctness of our results and assess their practical impact by analyzing prominent AC protocols. Our impossibility results naturally only offer necessary constraints for anonymity, but *not* sufficient conditions for the AC protocol. However, these necessary constraints for sender and recipient anonymity are crucial for understanding bi-directional anonymous communication. In fact, we find that several AC protocols in the literature are asymptotically close to the suggested constraints. Moreover, designers of new AC protocols can use our necessary constraints as guidelines for avoiding bad combinations of latency and bandwidth-overhead.

Our exercise in formally analyzing both protocols with user coordination and protocols without presents us with several tangible lessons: first, as our novel necessary constraints inherently are more lenient (they allow more anonymity with the same latency overhead and bandwidth overhead) our results thereby point future research on designing ACNs with re-

duced overhead in the direction of ACNs with user coordination, in particular with dynamic user coordination that relaxes the strict turn-by-turn scheduling of DC-nets and its several extensions. Second, we further contribute to the quest for optimal ACNs by clearly identifying the limits of our novel necessary constraints. Effectively, our necessary constraints raise open problems whose solutions would escape our results and could lead to ACNs that are very close to the universal necessary constraint.

Applications: optimal constructions of AC protocols. We use our derived necessary constraints as guidelines to construct optimal AC protocols. We try to construct AC protocols with overhead as close as possible to our derived lower bound on overhead to achieve strong anonymity. On the side, we also try to identify the building blocks that are fundamental towards achieving provable anonymity properties.

First, we construct a mixnet-based AC protocol *Streams* that tries to achieve anonymity as the cost of latency overhead. It offers provable anonymity to the users among all honest users of the system, while still offering horizontal scaling such that a node needs to cryptographically process only a tiny fraction of the total messages routed through the Streams network. Towards offering horizontal scaling we develop a novel super-node structure, which can be of independent interest to most mixing network systems.

Second, we construct a DC-net based AC protocol *OrgAn* that efficiently utilizes the user coordination technique to achieve strong anonymity against global passive adversaries with constant latency overhead. Utilizing a client/relay/server network topology, OrgAn uses key-homomorphic pseudorandom function and Netwon's power sums to effectively implement user coordination. OrgAn's cryptographic design allows it to overcome a significant problem with existing DC-net designs: unlike other DC-net designs OrgAn avoids frequent, interactive, slots and key agreement protocol.

Construction of a protocol that can effectively utilize both mixnet and user coordination techniques towards achieving anonymity still remains elusive.

1.2 Dissertation Outline

The dissertation is organized as follows: Chapters 3 and 4 presents the fundamental requirements (in terms of latency and bandwidth overhead) for anonymity, Chapters 5 and 6 presents constructions of AC protocols guided by the constraints derived in Chapters 3 and 4. In particular, Chapter 3 presents the first set of bounds for AC protocols that do not employ any kind of user coordination techniques. Then Chapter 4 presents the advantages of user coordination techniques, and derives the necessary constraints for anonymity when they are employed. In Chapter 5 we construct a mixnet-based protocol *Streams* guided by the restrictions provided by Chapter 3. Further, in Chapter 6 we construct a DC-net type protocol *OrgAn* that effectively utilizes user coordination. Finally, we summarize the dissertation in Chapter 7.

2. BACKGROUND

This chapter formally defines the indistinguishability-based anonymity notion for which we prove our necessary constraints, the notion of time, and the user behavior that we consider to prove our results. It also provides a brief overview of the proof technique that we use to prove our impossibility results.

2.1 Anonymity Definition

2.1.1 AnoA-Style Anonymity Definition

We define our anonymity notions with a challenge-response game following the AnoA definition [39], where the challenger simulates the protocol and the adversary tries to deanonymize users. The challenger $\texttt{Ch}(\Pi, \alpha, b)$ (formally defined in Fig. 2.1) allows the adversary to control user communication in the network, up to an uncertainty of one bit for challenges, and is parametric in the following parts: (i) the AC protocol Π to be analyzed, (ii) the so called *anonymity function* α, that describes the specific variant of anonymity such as sender anonymity, recipient anonymity and relationship anonymity, (iii) and the challenge bit b which determines the decision the challenger takes in challenge inputs from the adversary.

Given a security parameter η, we quantify the anonymity provided by the protocol Π simulated by $\texttt{Ch}(\Pi, \alpha, b)$ in terms of the advantage the probabilistic polynomial time (PPT) adversary $\mathcal{A}$ has in correctly guessing $\texttt{Ch}$'s challenge bit b. We measure this advantage in terms of indistinguishability of random variables additively, where the random variables in question represent the output of the interactions $\langle \mathcal{A} | \texttt{Ch}(\Pi, \alpha, 0) \rangle$ and $\langle \mathcal{A} | \texttt{Ch}(\Pi, \alpha, 1) \rangle$.

Definition 2.1.1 ((α, δ)-IND-ANO)**.** *A protocol* Π *is* (α, δ)-IND-ANO [1] *for the security parameter* η*, an adversary class* C*, an anonymity function* α *and a distinguishing factor* $\delta(\cdot) \geq 0$*, if for all PPT machines* $\mathcal{A} \in \mathrm{C}$*,*

$$\Pr\left[0 = \langle \mathcal{A} | \texttt{Ch}(\Pi, \alpha, 0) \rangle\right] \leq \Pr\left[0 = \langle \mathcal{A} | \texttt{Ch}(\Pi, \alpha, 1) \rangle\right] + \delta(\eta).$$

[1] ↑AnoA also allows a multiplicative factor ε; we use the simplified version with $\varepsilon = 0$, such that δ directly corresponds to the adversarial advantage.

For an anonymity function α, we say that a protocol Π provides *strong anonymity* [28,29] if it is $(\alpha, \delta) - \texttt{IND-ANO}$ with $\delta \leq neg(\eta)$ for some negligible function *neg*. If δ is instead *non-negligible* in η, then we say that Π provides *weak anonymity*.

Note that η does not measure the size of the anonymity set, but the computational limitation of the adversary.

Strong anonymity is relative to a strength η, which is bound to system parameters or analysis parameters such as the number of users or protocol parties, the latency overhead and the bandwidth overhead. These parameters typically increase as η increases, which improves the protocol's anonymity.[2] Anonymity in relation to η unifies a wide variety of possible analyses on how the anonymity bound changes with changing system parameters, and user numbers and behaviors. In particular, all our system and analysis parameters, such as the bandwidth overhead β, the latency ℓ, or the number of compromised parties c are actually functions in η. Each inequality on such parameters, e.g., $2\ell\beta < 1$, is an abbreviation for $2\ell(\eta)\beta(\eta) < 1$ for sufficiently large η.

Sender Anonymity. Sender anonymity characterizes the anonymity of users against a malicious server through the inability of the server (or some intermediary) to decide which of two *self-chosen* users have been communicating with the server. We borrow the sender anonymity α_{SA} definition from the AnoA framework [39], where α_{SA} selects one of two possible challenge users and makes sure that the users cannot be distinguished based on the chosen recipient(s) or message(s).

Definition 2.1.2 (Sender anonymity)**.** *A protocol* Π *provides* δ-sender anonymity *if it is* (α_{SA}, δ)-$\texttt{IND-ANO}$ *for* α_{SA} *as defined in Figure 2.1.*

Recipient Anonymity. Recipient anonymity characterizes that the recipient of a communication remains anonymous, even to observers that have knowledge about the sender in question. Similar to sender anonymity, we borrow the recipient anonymity α_{RA} definition from the AnoA framework, where α_{RA} selects one of two possible recipients for a message

[2]↑In some analyses, individual parameters may reduce with increasing η, such as the bandwidth overhead per user, as the other parameters, such as the number of users, increase.

and makes sure that the recipients cannot be distinguished based on the chosen sender(s) or message(s).

Definition 2.1.3 (Recipient anonymity)**.** *A protocol* Π *provides* δ-recipient anonymity *if it is* (α_{RA}, δ)-IND-ANO *for* α_{RA} *as defined in Figure 2.1.*

We omit the detailed technical notation of the anonymity functions in the following sections, and write $\Pr[0 = \mathcal{A} \mid b = i]$ instead of $\Pr[0 = \langle \mathcal{A} | \mathtt{Ch}(\Pi, \alpha_{SA}, i) \rangle]$.

Adaptive AnoA Challenger $\mathtt{Ch}(\Pi, \alpha, b)$

Upon message $(\mathtt{Input}, u, R, m)$**:** *RunProtocol*$(u, R, m)$

Upon message $(\mathtt{Chall}, u_0, u_1, R_0, R_1, m)$**:**

Compute $(u^*, R^*) \leftarrow \alpha(u_0, u_1, R_0, R_1, b)$
RunProtocol(u^*, R^*, m)

RunProtocol(u, R, m)**:**

Run Π on $r = (u, R, m)$ and forward all messages that are sent by Π to the adversary $\mathcal{A}$ and send all messages by the adversary to Π.

$\alpha_{SA}(u_0, u_1, R_0, R_1, b) = (u_b, R_0)$

$\alpha_{RA}(u_0, u_1, R_0, R_1, b) = (u_0, R_b)$

Figure 2.1. Adaptive AnoA Challenger

2.1.2 On the meaning of η

In our analyses we tie η to system parameters such as the number of parties K, the number of compromised parties c, the latency overhead ℓ, the bandwidth overhead B, the number of users N, etc.; we explicitly describe the relationship between η and these parameters for the cases we consider. The system parameters don't have to increase with η necessarily. In some cases, parameters may decrease as η increases, for example, the bandwidth overhead B

might decrease as the latency overhead increases, or the ratio of compromised (or honest!) parties might decrease.

Note that if an AC protocol has strong anonymity, it is secure under continual observation (e.g., for streams of messages or usage over a longer time period) and formally, η limits the number of observations.

On anonymity sets and strong anonymity. Strong sender anonymity lets the adversary freely choose the pair of challenge senders and requires that the AC protocol's behavior is indistinguishable to the adversary. Hence, strong sender anonymity corresponds to the full anonymity set (see [39, Lemma 7]) that encompasses all users. Sender anonymity for (non-full) anonymity sets would result in restricting the adversary in Definition 2.1.1 to choose the pair of challenger senders from the same anonymity set.

2.1.3 Game Setup

Let $\mathcal{S}$ be the set of all senders, $\mathcal{R}$ be the set of all recipients, and P be the set of protocol parties that participate in the execution of the protocol (like relays/mix-nodes in Tor/mix-nets, for DC-net or P2P mixing users and protocol parties are the same). We consider a system of total $|\mathcal{S}| = \mathsf{N}$ senders. For *sender anonymity*, we need only a single element in $\mathcal{R}$, while for recipient anonymity we only need one in the set $\mathcal{S}$. We allow the adversary (for sender anonymity) to set the same entity (say R) as the recipient of all messages, and expect R to be compromised by the adversary. The adversary uses a challenge (as defined in Figure 2.1) of the form $(u_0, u_1, R, \_, m_0)$, where $u_0, u_1 \in \mathcal{S}$, for our sender anonymity game.

We consider a completely connected topology, which means any party can send a message directly to any other party. We assume a standard (bounded) synchronous communication model as in [15,35,42,43], where a protocol operates in a sequence of communication rounds.[3] In each round, a party performs some local computation and then sends messages (if any) to other parties through an authenticated link. By the end of the round, every party receives all messages sent by other parties to her in the same round. With our focus on computing

[3]↑While a time-sensitive model [44] would be more accurate, e.g., for low-latency protocols like Tor [13], such a model would only strengthen the attacker. As we present necessary constraints, our results also hold for the more accurate setting.

lower bounds, our model abstracts from the time the computations at the node take and also the length of the messages. Nevertheless, as we are interested in quantifying the communication/bandwidth overhead, unlike [15,35,43], we do not assume that the parties have access to ready-made broadcast communication channels; Parties are expected to communicate with each other to implement broadcast features [42, 45]. We stress that using an asynchronous communication model offers more capabilities to the attacker, and thus, our impossibility results for this synchronous model naturally apply to an asynchronous model as well.

We define the *latency overhead* ℓ as the number of rounds that a message can be delayed by the protocol before being delivered. We define the *bandwidth overhead* β as the number of noise messages per user that the protocol can create in every round (i.e., the dummy message rate) and we do not restrict the time these noise messages reside within the protocol.

We consider two types of *global passive* adversaries: Our *non-compromising* adversaries (which model network-level eavesdroppers) can observe all communication between all protocol parties, but do not compromise any party of the AC protocol except the recipient R. We say that the AC protocol is *non-compromised*. Our strictly stronger *partially compromising* adversaries (which model hacking and infiltration capabilities) can additionally compromise some of the AC parties in the setup phase of the game to obtain these parties' mapping between the input messages and output messages during the protocol's runtime. In the case of partially compromising adversaries, we say that the AC protocol is *partially compromised*.

2.2 User Message Distributions, Communication Rounds, Bandwidth Overhead, and Latency

2.2.1 Time

We use a round-based definition of time in which we assume that all protocol parties work in synchronized rounds. In each round, a party can send packets to other parties that will receive the packets at the end of the round (and can then send them on in the next round). We allow, but abstract away from any cryptographic operations locally performed on these packets and we don't consider the computation time required for such operations:

independently of the cryptographic operations performed, a packet is always ready for being sent in the round after it arrived.

2.2.2 Latency and Bandwidth Overhead

We define the latency overhead ℓ of a protocol as the number of rounds that pass between the round in which a message is scheduled for being (originally) sent by a user u and the round it is received (and potentially reconstructed) by a recipient R. We define the *bandwidth overhead* B as the number of noise messages that the protocol can create for every real message. We additionally define β as the number of noise messages per user that the protocol can create in every round, i.e., the dummy message rate.

2.2.3 User Message Distributions

The user message distribution describes how we select which user sends messages at which point in time. This is crucial for anonymity as it defines which users are active and how often they participate in the protocol.

We consider two kinds of user message distributions in our anonymity games and both of them assume an N sized set $\mathcal{S}$ of users that want to send messages. In both cases, the adversary can choose any two senders $u_0, u_1 \in \mathcal{S}$. However, the time and method by which they actually send messages differs:

- In the *synchronized* user message distribution U_B, the users globally synchronize who should send a message at which point in time. We assume that each user wants to send exactly one message. Consequently, we choose a random permutation of the set of users $\mathcal{S}$ and the users send messages in their respective round. In every single round out of a total of N rounds exactly one user sends a message. Since the users globally synchronize their sending of messages, we allow the protocol to also globally decide on the bandwidth overhead it introduces. Note that here the requirements are identical to those of the Bulk protocol in [35].

- In the *unsynchronized* user message distribution U_P, each of the N users wants to send messages eventually and we assume that each user locally flips a (biased) coin every round to decide whether or not to send a message. In this case we define the bandwidth overhead as an increased chance of users sending messages. Since the protocol does not globally synchronize the input messages, for noise messages also we allow the users to decide it locally and send noise messages with a certain probability.

The synchronized user message distribution can be seen as a control group that is predictable and thus fairly protocol friendly. Protocols following DC-nets tend to use such a synchronization (to ensure that messages from a sender can actually be reconstructed). Our results show that many interesting cases are the same for this predictable user distribution U_B and for the unsynchronized U_P.

2.3 Brief Overview of the Proof Technique

To show that there is not (and, in fact, cannot be) a protocol that provides strong anonymity without a significant bandwidth overhead and/or latency overhead, we need to capture all possible protocols and show that each of them is vulnerable to attacks. In general, we derive our results in five main steps.

First, we formally define a protocol model that serves the purpose of formally specifying which protocols are considered. The (impossibility) results are valid for protocols that can be expressed in the given protocol model.

Second, we define a concrete adversary $\mathcal{A}_{paths}$, that uses a well established strategy: upon recognizing the challenge message (as soon as it reaches a receiver) $\mathcal{A}_{paths}$ constructs the possible paths this message could have taken through the network, and tries to identify the user who has sent the message.

Next, given the concrete adversary $\mathcal{A}_{paths}$, we identify a necessary invariant that any protocol has to fulfill in order to provide anonymity. Intuitively: *both challenge users chosen by the adversary must be active (i.e., send at least one message) before the challenge message reaches the recipient, and it must be possible for these messages to meet in at least one honest party along the way.*

Next, we propose an ideal protocol Π_{ideal} that is optimal in terms of satisfying the invariant: The probability that Π_{ideal} fulfills the necessary invariant is at least as high as for any protocol within our model (limited by the same constraints for β and ℓ). Thus, no protocol can be better at winning against $\mathcal{A}_{paths}$ than Π_{ideal} is in satisfying the necessary invariant.

Finally, we calculate a bound δ on the probability of Π_{ideal} to satisfy the necessary invariant. By calculating δ, we obtain a lower bound (of δ) on the adversarial advantage against all protocols within our model.[4]

These steps let us conclude that any protocol (which cannot be better than the idealized one) is vulnerable to this specific adversary. Our specific adversary is fairly simple and possibilistic. There are more sophisticated adversaries that, e.g., take the expected distribution for each sender into account. To such adversaries the protocols are potentially even more vulnerable; hence, our results might be untight.

As *non-compromising* adversaries are a subset of *partially compromising* adversaries, our proof technique for the former is a simplified case of the latter.

[4]↑ $\mathcal{A}_{paths}$ is a possible adversary against all protocols within our model. If $\mathcal{A}_{paths}$ wins whenever the invariant is not satisfied and our ideal protocol Π_{ideal} (bounded by β and ℓ) is the best protocol for satisfying the invariant, then $\mathcal{A}_{paths}$ will also have an advantage of at least δ against any protocol within our model (that is also bounded by β and ℓ). Thus, our bound for δ describes a lower bound on the adversarial advantage against any protocol within the model, while against particular protocols there can be other adversaries (in the same adversary class) with an even higher advantage.

3. FIRST ANONYMITY TRILEMMA

In this chapter we present our first set of impossibility results for all possible AC protocols except protocols that employ user coordination techniques. These results demonstrate that AC protocols without user coordination have a fundamental necessary constraint (in terms of latency and bandwidth overhead) to achieve strong anonymity against global network level adversaries. We start with the formal protocol model that captures all the AC protocols in the considered protocol class.

ℓ	Latency overhead for every message
β	number of noise packets for every user per round
B	number of noise packets per real message
p	Probability to send a message per user per round
p	Probability to send a real message per user per round
K	Number of (internal) protocol parties
c	Number of compromised protocol parties
N	Number of online users (that may send messages)
δ	Adversarial advantage in the anonymity game
Π	A protocol. $\Pi \in M$: Π is within our model
η	The security parameter
ϵ	A (very small, but non-negligible) function

Figure 3.1. Notation

3.1 A Protocol Model for AC Protocols

An AC protocol allows any user in the set of users $\mathcal{S}$ to send messages to any user in $\mathcal{R}$, via a set of anonymizing parties P. We define protocols that are under observation of an eavesdropping adversary $\mathcal{A}$ that may have compromised a set of c parties $\mathsf{P_c} \subseteq \mathsf{P}$ and that furthermore observes the communication links between any two parties, including users.

Technically, whenever a party $P_1 \in \mathsf{P} \cup \mathcal{S}$ sends a message to another party $P_2 \in \mathsf{P} \cup \mathcal{R}$, the adversary is able to observe this fact together with the current round number. However, we assume the protocol applies sufficient cryptography, s.t., the adversary can not read

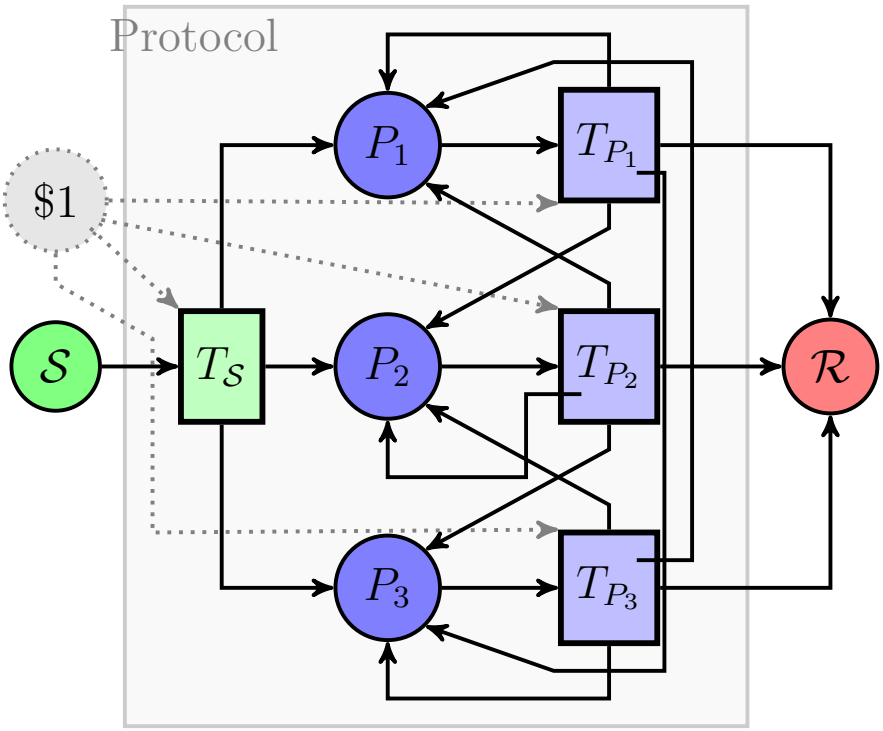


Figure 3.2. Petri net of an AC protocol with $\mathsf{K} = 3$ parties.

the content of any message except the messages sent to the malicious recipient. Thus, the adversary can only recognize the challenge message when it reaches the recipient.

For an actual protocol, the sets $\mathcal{S}$, $\mathcal{R}$, and P might not be mutually exclusive [11, 12, 15]. Since we have only one malicious party in $\mathcal{R}$, and the content of a message can only be read when it reaches its final recipient, we consider $\mathcal{R}$ to be mutually exclusive from $\mathcal{S} \cup \mathsf{P}$ for the purpose of simplicity.

With the above preliminaries in mind, we shall now formally define our generic AC protocol using a Petri net model.

3.1.1 Protocol Model

We model any AC protocol with K parties by a timed colored Petri net [40, 41, 46] M, consisting of places $\mathcal{S}$ for the users, $P_1, \ldots, P_{\mathsf{K}}$ symbolizing the protocol parties, \$1 for randomness and R for recipients of messages, and colored tokens m symbolizing the messages (real or noise) sent by clients or protocol parties, and transitions $T_{\mathcal{S}}$ for inserting messages

into the network and $T_{P_1}, \ldots, T_{P_K}$ as functions for sending the messages from one party to another. The structure of the Petri net with its places, tokens and transitions remains the same for every AC protocol. However, the implementation of the guards within the transitions is different for different protocols: protocols can choose to which party messages are to be sent next and whether they should be delayed. Protocols in M are oblivious to the challenge message or the challenge users – and so is the implementation of the guards within the transitions. We refer to Figure 3.2 for a graphical depiction of Petri net model M.

Definition 3.1.1 (Colored token)**.**

A colored token *is represented by the tuple* $m = \langle \mathsf{msg}, \mathsf{meta}, \mathsf{t_r}, \mathsf{ID_t}, \mathsf{prev}, \mathsf{next}, \mathsf{ts} \rangle$*, where,*

- msg *is the content of the message,*
- meta *is the internal protocol meta-data for this message,*
- $\mathsf{t_r}$ *is the time the message can remain in the network,*
- $\mathsf{ID_t}$ *is a new unique ID generated by each transition for each token by honest parties; dishonest parties instead keep* $\mathsf{ID_t}$ *untouched to allow the adversary to link incoming and outgoing messages,*
- prev *is party/user that sent the token and* next *is the user/party that receives the token.*
- *Finally,* ts *is the time remaining for the token to be eligible for a firing event (a feature of timed Petri net). Here,* ts *either describes when new messages are introduced into the Petri net or is set to the next round, such that messages can be processed in every round as soon as they enter the network.*

The four fields $\mathsf{ID_t}, \mathsf{prev}, \mathsf{next}, \mathsf{ts}$ are public, and are visible to the adversary. The remaining three fields msg, meta and $\mathsf{t_r}$ in a token are private and can not be observed by the adversary, with the exception that msg can be observed when a message reaches its destination, i.e, is received by a recipient. [1]

[1]↑For sender anonymity we only consider one recipient and thus, for simplicity, do not need to specify the recipient in the token. For recipient anonymity, the colored token additionally has a private field for the recipient.

Formally, we introduce a set Tokens, that is initially empty and in which we collect the pair (t, r), where t is a copy of a token and r the round number in which the token was observed.

Places. Any AC protocol with K parties $\mathsf{P} = \{P_1, \ldots, P_{\mathsf{K}}\}$ consists of the following places:

- $\mathcal{S}$: A token in $\mathcal{S}$ denotes a user message (real or noise) which is scheduled to enter the network after ts rounds.
- \$1: This place is responsible for providing randomness. Whenever a transition picks a token from this place, the transition basically picks a random value.
- P_i with $P_i \in \mathsf{P}$: A token in P_i denotes a message which is currently held by the party $P_i \in \mathsf{P}$.
- R: A token in R denotes a message which has already been delivered to a recipient.

Transitions in the Petri net model M

$\mathbf{T_S}$ on tokens $q = \langle \mathsf{msg}, \_, \_, \_, u, \_, \mathsf{ts} \rangle$ from $\mathcal{S}$ and \$ from \$1:

$(P_i, \mathsf{meta}) = f_\Pi(q, \$)$; $\mathsf{ID_t}$ = a fresh randomly generated ID
r = current round; $t = \langle \mathsf{msg}, \mathsf{meta}, \ell, \mathsf{ID_t}, u, P_i, 1 \rangle$
if $P_i = R$ **then** $\mathsf{Tokens} = \mathsf{Tokens} \cup (\langle \mathsf{msg}, \_, \_, \mathsf{ID_t}, u, P_i, 1 \rangle, r)$
else $\mathsf{Tokens} = \mathsf{Tokens} \cup (\langle \_, \_, \_, \mathsf{ID_t}, u, P_i, 1 \rangle, r)$
Output: token t at P_i

$\mathbf{T_{P_i}}$ on tokens $q = \langle \mathsf{msg}, \_, \mathsf{t_r}, \mathsf{ID_t}, \_, P_i, \mathsf{ts} \rangle$ from P_i, \$ from \$1:

$(P, \mathsf{meta}) = f_\Pi(q, \$)$; r = current round
if $\mathsf{t_r} - 1 = 0$ **then** $P = R$
if P_i is honest **then** $\mathsf{ID_t}$ = a fresh randomly generated ID
else if P_i is compromised **then** $\mathsf{ID_t} = \mathsf{ID_t}$
$t = \langle \mathsf{msg}, \mathsf{meta}, \mathsf{t_r} - 1, \mathsf{ID_t}, P_i, P, 1 \rangle$
if $P_i = R$ **then** $\mathsf{Tokens} = \mathsf{Tokens} \cup (\langle \mathsf{msg}, \_, \_, \mathsf{ID_t}, P_i, P, 1 \rangle, r)$
else $\mathsf{Tokens} = \mathsf{Tokens} \cup (\langle \_, \_, \_, \mathsf{ID_t}, P_i, P, 1 \rangle, r)$
Output: token t at P

f_Π: The code for this function is provided by protocol Π. It decides to which party the message is sent next, as well as the content of the meta field in the token.

Figure 3.3. Transitions in the Petri net model M

Transitions. As part of the *initial configuration*, the challenger populates $\mathcal{S}$ on behalf of the protocol. All other places are initially empty. The transitions then consumes token from

one place and generate tokens in other places, to modify the *configuration* of the Petri net. The event of consumption of a token from one place by a transition and generation of a new token represents the movement of a message from one party to another. We define the following transitions (we refer to Figure 4.1 for the pseudocodes of the transitions):

- $T_{\mathcal{S}}$: takes a token $\langle \mathsf{msg}, \_, \_, \_, u, \_, \mathsf{ts} \rangle$ from $\mathcal{S}$ and a token from \$1 to write $t = \langle \mathsf{msg}, \mathsf{meta}, \ell, \mathsf{ID_t}, u, P_i, \mathsf{ts} = 1 \rangle$ to P_i; the values of i and meta are decided by the AC protocol.
- T_{P_i}: takes a token $\langle \mathsf{msg}, \mathsf{meta}, \mathsf{t_r}, \mathsf{ID_t}, \_, P_i, \mathsf{ts} \rangle$ from P_i and a token from \$1 to write $t = \langle \mathsf{msg}, \mathsf{meta}, \mathsf{t_r} - 1, \mathsf{ID_t}, P_i, P, 1 \rangle$ to P. If P_i is an honest party $\mathsf{ID_t}$ is freshly generated, but if P_i is a compromised party $\mathsf{ID_t} = \mathsf{ID_t}$. The place $P \in \{P_1, \ldots, P_{\mathsf{K}}\} \cup \{R\}$ and meta are decided by the AC protocol, with the exception that if $\mathsf{t_r} = 0$, P always is R.

In either case, the transition also adds an element (t, r) to the set Tokens, where r is the current round number and t is a copy of the respective (new) token t, with the fields meta and $\mathsf{t_r}$ are removed. If the place where t was written to is not $\mathcal{R}$, then additionally the field msg is removed.

3.1.2 Game Setting

We use the sender anonymity notion from the AnoA framework [39] as mentioned in Chapter 2. To increase readability, we summarize the AnoA definition and explicitly write down how the sender anonymity notion α_{SA} works and how the wrapper around the protocol is executed. We consider the following game between a PPT adversary $\mathcal{A}$ and an honest challenger $\mathtt{Ch}(\Pi, \alpha_{SA}, b)$:

- $\mathcal{A}$ compromises up to c parties from P.
- $\mathcal{A}$ chooses two distinct users u_0 and u_1 as challenge users. $\mathcal{A}$ sends a challenge message $(\mathtt{Chall}, u_0, u_1, R, R, m^*)$ for those chosen users.
- $\mathtt{Ch}$ then runs protocol Π on (u_b, R, m^*). Π is executed in two parts, Π_{wrapper} and Π_{core}, as described below. (We refer to Figure 4.2 for the pseudocode of Π.)
- First, Π_{wrapper} generates tokens following the user distribution and embeds the challenge message (u_b, R, m^*) in the tokens. Π_{wrapper} adds all the tokens to the place

$\mathcal{S}$. However, Π_{wrapper} does not pass any information about the challenge user or the challenge message to Π_{core}.

- We then allow Π_{core} to add noise tokens to $\mathcal{S}$, limited by the protocol's restrictions on bandwidth overhead. After that, Π_{core} runs the petri net with protocol specific implementation of the transitions.
- For each element in Tokens, $\mathcal{A}$ can see the round number as well as the public parts of the token ($\mathsf{ID_t}, \mathsf{prev}, \mathsf{next}, \mathsf{ts}$), but the private parts ($\mathsf{msg}, \mathsf{meta}, \mathsf{t_r}$) are not communicated to the adversary (c.f. Figure 4.1). However, when the field next is the recipient, the msg field is not obfuscated.
- The goal of the adversary is to deanonymize the sender of the challenge message, i.e., to learn whether the challenge message was sent by u_0 or by u_1. The interaction between $\mathtt{Ch}$ and $\mathcal{A}$ ends as soon as $\mathcal{A}$ makes a guess.

Run protocol Π **on** $r = (u^*, R^*, m^*)$

Π **on** $r = (u^*, R^*, m^*)$ **and user distribution** U**:**

$I_U \leftarrow \Pi_{\text{wrapper}}(r, U)$,
where I_U is a set and each element in I_U is a tuple (u, R, m, ts).
Run $\Pi_{\text{core}}(I_U, U, \beta)$.

$\Pi_{\textbf{wrapper}}$ **on** $r = (u^*, R^*, m^*)$ **and user distribution** U**:**

generate a set I_U following U,
where I_U is a set and each element in I_U is a tuple (u, R, m, ts).
$e = (u, R, m, \mathsf{ts}) \xleftarrow{\$} I_U$ such that $u = u^* \wedge R = R^*$.
if e exists **then**
  $I_U \leftarrow \{(u^*, R^*, m^*, \mathsf{ts})\} \cup I_U \setminus \{e\}$
for each element $e = (u, R, m, \mathsf{ts})$ in I_U **do**
  Add a token $t = \langle m, \_, \_, \_, u, \_, \mathsf{ts} \rangle$ in the place $\mathcal{S}$.
Output: I_U

$\Pi_{\textbf{core}}$ **on** I_U**,** U**,** β**:**

Add tokens in the place $\mathcal{S}$ within the limit of β noise messages per user per round.
Run the petri net with the protocol specific implementation of f_Π.

Figure 3.4. Description of protocol Π

Validity of the Protocol Model. Protocols in the above protocol model behave as expected (more details in Lemma 1 in Section 3.A). We show in Lemma 1 that the protocols indeed have a bandwidth overhead of β and a latency overhead of ℓ. For every message that is sent from one party in $\mathcal{S} \cup \mathsf{P}$ to another party in $\mathsf{P} \cup \mathcal{R}$, the adversary learns the time, the sender, and the receiver. When a message leaves the network, the attacker learns whether it was the target (i.e., the challenge) message. The attacker also learns the mapping between the input and output messages of compromised parties. For recipient anonymity, the attacker instead learns whether a message is the target (i.e., challenge) message after it leaves the sender.

3.1.3 Expressing Protocols

Our protocol model M allows the expression of any AC protocol with very few, esoteric exceptions. Here we explain how our model can capture different techniques that are used in different AC protocols.

Mix networks can be naturally embedded into our model, in particular any stop-and-go mix [34] that uses discrete distribution and even AC protocols with specialized path selection algorithms [47,48]. For the sake of our necessary constraints, low-latency protocols (with time-bounded channels) that are not round-based (e.g., Tor [49]) can be expressed in a round-based variant, since it only strengthens the protocols anonymity properties. This section illustrates embedding techniques into our model for some other kinds of protocols, but a much larger variety of protocols can be expressed in our model.

Users as protocol parties. In peer-to-peer protocols where the users act as type of relays. any noise sent by users counts into the bandwidth overhead of the protocol (we will see in Claim 2 that noise sent by nodes that are not users can be treated differently). Whenever a user wants to send a message it should use the transition $T_{\mathcal{S}}$, but when it acts as a relay it should use the transition T_{P_i}.

Splitting and Recombining Messages. We model protocols that split and later recombine messages by declaring one of the parts as the main message and the other parts as noise, which may count into the bandwidth overhead. This declaration is mainly required for

the analysis, i.e., for evaluating the success of the adversary and for quantifying the amount of noise messages introduced by the protocol. We do not restrict the strategy by which the protocol decides which message is "the main share" (i.e., the message that is sent on) and which is "an additional share" (i.e., a fresh noise message). A more complex scenario involves threshold schemes in which a smaller number of shares suffices for reconstructing the message and in which some shares are dropped randomly. In such cases we consider the protocol to decide beforehand which of the constructed shares will be dropped later and to declare one of the remaining shares the "main share".

Broadcasting Messages. If the protocol chooses to copy or broadcast messages to several receivers, we consider the copy sent to the challenge receiver to be the main message and copies sent to other receivers to be noise (which, if the copies are created by nodes that are not users, will not count into the bandwidth overhead).[2]

Private Information Retrieval. In schemes based on private information retrieval we require that the receiver retrieves the information sufficiently fast (within the latency limit). Otherwise, our method is similar to the broadcasting of messages: the receiver of interest will retrieve the main message, whereas other receivers will retrieve copies that are modeled as noise.

Excluded Protocols. For this work we exclude protocols that cannot guarantee the delivery of a message within the given latency bound (except if this occurs with a negligible probability). Moreover, we cannot easily express the exploitation of side channels to transfer information, e.g., sending information about one message in the meta-data of another message, or sending bits of information by not sending a message.

In this chapter, we do not consider protocols with user coordination, which we consider in the next chapter.

[2]↑We note that in some cases, where users act as nodes and broadcast messages to other users, our quantification of the bandwidth overhead might be a bit harsh. If the group of users to which the broadcast will be sent is known in advance (i.e., if messages are broadcast to all users or to pre-existing groups of users), we can allow the protocol to use a single receiver for these messages instead.

3.1.4 Construction of a Concrete Adversary

Given two challenge users u_0 and u_1 and the set of observed tokens $(t, r) \in \mathsf{Tokens}$, where t is the token and r the round in which the token was observed, an adversary can construct the sets S_j (for $j \in \{0, 1\}$). Assume the challenge message arrives at the receiver R in a round r. We construct possible paths of varying length k, s.t., each element $p \in S_j$ represents a possible path of the challenge message starting from u_j ($j \in \{0, 1\}$) and the challenge message then arrives at the recipient R in round $r_k = r$. With challenge bit b, S_b cannot be empty, as the actual path taken by the challenge message to reach R has to be one element in S_b.

$$
\begin{aligned}
S_j = \{p = (t_1.\mathsf{prev}, \ldots, t_k.\mathsf{prev}, t_k.\mathsf{next}) : \\
((t_1, r_1), \ldots, (t_k, r_k)) \in \mathsf{Tokens} \text{ s.t.} \\
t_1.\mathsf{prev} = u_j \wedge t_k.\mathsf{next} = R \wedge t_k.\mathsf{msg} = \texttt{Chall} \wedge k \leq \ell \\
\wedge \forall_{i \in \{1,\ldots,k-1\}} (t_i.\mathsf{next} = t_{i+1}.\mathsf{prev} \wedge r_{i+1} = r_i + 1 \\
\wedge (\ \exists t_{i+1} : (t_{i+1}, r_{i+1}) \in \mathsf{Tokens}\ \wedge t_{i+1}.\mathsf{prev} = t_i.\mathsf{next} \\
\wedge t_{i+1}.\mathsf{ID_t} = t_i.\mathsf{ID_t}) \Rightarrow t_{i+1} = t_{i+1})\}
\end{aligned}
$$

Definition 3.1.2 (Adversary $\mathcal{A}_{paths}$)**.** *Given a set of users $\mathcal{S}$, a set of protocol parties P of size K, and a number of possibly compromised nodes c, the adversary $\mathcal{A}_{paths}$ proceeds as follows: 1. $\mathcal{A}_{paths}$ selects and compromises c different parties from P uniformly at random. 2. $\mathcal{A}_{paths}$ chooses two challenge users $u_0, u_1 \in \mathcal{S}$ uniformly at random. 3. $\mathcal{A}_{paths}$ makes observations and, based upon those, constructs the sets S_0 and S_1. For any $i \in \{0, 1\}$, if $S_i = \emptyset$, then $\mathcal{A}_{paths}$ returns $1 - i$. Otherwise, it returns 0 or 1 uniformly at random.*

$\mathcal{A}_{paths}$ thus checks whether both challenge users *could have* sent the challenge message. We explicitly ignore differences in probabilities of the challenge users having sent the challenge message, as those probabilities can be protocol specific. Naturally, when $\mathsf{c} = 0$, $\mathcal{A}_{paths}$ represents a *non-compromising* adversary; but when $\mathsf{c} \neq 0$, $\mathcal{A}_{paths}$ is *partially compromising.*

3.1.5 Protocol Invariants

We now investigate the robustness of protocols against our adversary. We define an invariant that, if not satisfied, allows $\mathcal{A}_{paths}$ to win against any protocol. Moreover, we present a protocol that maximizes the probability of fulfilling the invariant.

Necessary invariant for protocol anonymity. It is necessary that at least both challenge users send messages in one of the ℓ rounds before the challenge message reaches the recipient, as otherwise there is no way both of them could have sent the challenge message. Moreover, on the path of the actual challenge message, there needs to be at least one honest (uncompromised) party, as otherwise the adversary can track the challenge message from the sender to the recipient (S_b will have exactly one element and S_{1-b} will be empty). Those two conditions together form our *necessary protocol invariant.*

Invariant 1. *Let u_0 and u_1 be the challenge users; let b be the challenge bit; and let t_0 be the time when u_b sends the challenge message. Assume that the challenge message reaches the recipient at r. Assume furthermore that u_{1-b} sends her messages (including noise messages) at $V = \{t_1, t_2, t_3, \ldots, t_k\}$. Now, let $T = \{t : t \in V \wedge (r - \ell) \leq t < r\}$. Then,*

(i) the set T is not empty, and

(ii) the challenge message passes through at least one honest node at some time t such that, $t \in \{\min(T), \ldots, r-1\}$.

Claim 1 (Invariant 1 is necessary for anonymity)**.** *Let Π be any protocol $\in M$ with latency overhead ℓ and bandwidth overhead β. Let u_0, u_1, b and T be defined as in Invariant 1. If Invariant 1 is not satisfied by Π, then our adversary $\mathcal{A}_{paths}$ as in Definition 4.3.1 wins.*

Proof. We distinguish two cases, depending on T: either T is empty, or T is non-empty.

If the set T is empty, then S_{1-b} is empty as well. However, by construction of our protocol model, the set S_b is always non-empty. Consequently, the adversary $\mathcal{A}_{paths}$ will output b and thus win with probability 1. If T is not empty, the following cases can occur:

1. The challenge message never passes through an honest node: In this case, the field $\mathsf{ID_t}$ of the message never changes for the tokens. By definition of the sets S_j, the tokens

can only be combined if either there is no corresponding token with the same value for $\mathsf{ID_t}$, or by extending the path by exactly this token. Thus, S_b will have exactly one element, and S_{1-b} will be an empty set, and consequently $\mathcal{A}_{paths}$ wins.

2. The challenge message passes through one or more honest nodes at times t, such that $t < \min(T)$, but not afterwards. Following the same reasoning as above, we see that paths before $\min(T)$ can be ambiguous, but none of them leads to u_{1-b}. Hence, S_b can have multiple elements, but S_{1-b} will still be an empty set. Thus, $\mathcal{A}_{paths}$ wins.
3. The challenge message passes through an honest node at time t with $t \geq min(T)$. In this case, the invariant is true.

In all of the above mentioned cases either the invariant is true, or the adversary wins with probability 1. □

We next claim that it suffices to consider noise messages sent by users that also remain within the system for at most ℓ rounds, i.e., noise messages that follow the same rules as real messages. Note that we consider every new message originating from any user's client as a fresh noise message.

Claim 2 (Internal noise does not influence Invariant 1)**.** *Any message not originating from an end user $u \in \mathcal{S}$ does not influence the probability for Invariant 1 being true. Moreover, noise messages do not contribute to the probability for Invariant 1 being true after they stayed in the network for ℓ rounds.*

Proof. Let u_0, u_1 be the challenge users and let b be the challenge bit and let r be the round in which the challenge message is delivered to the recipient. We discuss both parts of the invariant separately:

(i) The set T is not empty. Since by definition, T is the set of messages sent by u_{1-b}, messages originating in any party not in $\mathcal{S}$ do not influence T. Moreover, any message sent by u_{1-b} in a round previous to $r - \ell$ does not influence T either. Thus, noise messages staying in the protocol for more than ℓ rounds, does not improve the probability of T being not empty.

(ii) The challenge message passes through at least one honest node at some time t such that, $t \in \{\min(T), \ldots, r-1\}$. Obviously this second part of the invariant does not depend on any noise message. □

Consequently, noise introduced by u in P but not in $\mathcal{S}$ do not modify the probability to fulfill Invariant 1. We henceforth consider noise messages as a protocol input.

3.1.6 Ideal Protocol

We construct a protocol Π_{ideal} that maximizes the probability of fulfilling Invariant 1. Claim 1 shows that for any protocol in our model $\mathcal{A}_{paths}$ wins whenever Invariant 1 does not hold. Thus, an upper bound on the probability that Π_{ideal} satisfies Invariant 1 yields an upper bound for all these protocols.

Given the set of all protocol parties $\mathsf{P} = \{P_0, \ldots, P_{\mathsf{K}-1}\}$ of size K, the strategy of Π_{ideal} is as follows: in a round r, Π_{ideal} delivers all messages scheduled for delivery to a recipient. All other messages (including the messages that enter Π_{ideal} in round r) are sent to the protocol party P_i with $i = r \mod \mathsf{K}$. For every message that enters the protocol, Π_{ideal} queries an oracle O for the number of rounds the message should remain in the protocol. We define the following events:

- $u.\text{sent}(x, y)$: user u has sent at least one message within rounds from x to y. For a single round we use $u.\text{sent}(x)$.
- $\text{Cmpr}(x)$: $\mathcal{A}_{paths}$ has compromised the next x consecutive parties on the path.
- $\neg H$: NOT of event H.

Given a message sent at t_0 by sender x, and delivered to the recipient at $(t_0 + t)$, we define P_t for sender $v \in \mathcal{S} \setminus \{x\}$:

$$\begin{aligned} P_t = &\sum_{j=r-\ell}^{t_0} \Pr\left[v.\text{sent}(j) \wedge \neg v.\text{sent}(j+1, t_0)\right] \cdot \Pr\left[\neg\text{Cmpr}(t)\right] \\ &+ \sum_{j=t_0+1}^{r} \Pr\left[v.\text{sent}(j) \wedge \neg v.\text{sent}(r-\ell, j-1)\right] \cdot \Pr\left[\neg\text{Cmpr}(r-j)\right] \end{aligned}$$

When $v = u_{1-b}$, and the message is the challenge message, P_t is the probability of fulfilling Invariant 1, for the strategy above. For each message, oracle O chooses an *optimal* t that

maximizes the expectation of P_t over all users. Due to the over-approximation with this (not realizable) oracle, the resulting protocol is optimal w.r.t. Invariant 1 (refer to Claim 3).

Claim 3 (Ideal protocol is ideal for the invariant)**.** *Against the given adversary $\mathcal{A}_{paths}$, Π_{ideal} satisfies Invariant 1 with probability at least as high as any other protocol in M.*

Proof. We want to prove our claim by contradiction. Suppose, Π_{ideal} is not the best protocol. That means, there exists a protocol Π_{new}, which satisfies Invariant 1 with a higher probability than Π_{ideal}, against the adversary $\mathcal{A}_{paths}$.

Now we construct a new protocol Π_{hybrid}, which exactly follows the strategy of Π_{ideal} with one exception: for a given message Π_{hybrid} selects the time delay t same as Π_{new}, instead of querying it from oracle O. Suppose, the challenge message is delivered to the recipient at round r. Given the set $\{\min(T), \ldots, r-1\}$, the ideal strategy for ensuring that at least one honest party is on the path of the challenge message is to ensure that as many distinct parties as possible are on this path. Also, given the time delay t, the value of $min(T)$ is independent of the protocol, since protocols in M are oblivious to the challenge users and the challenge message. Hence, Π_{hybrid} has a probability of satisfying Invariant 1 at least as high as Π_{new}.

Now, if we compare Π_{hybrid} and Π_{ideal}: they follow the same strategy. But Π_{ideal} picks the time delay t for any message from oracle O such that t is *optimal*. The time delay t can be picked for each message independent of the time delays of other messages. Hence, the value of t received from oracle O for the challenge message is optimal. Hence, Π_{ideal} satisfies Invariant 1 with probability at least as high as Π_{hybrid}. Thus, Π_{new} does not satisfy Invariant 1 with a higher probability than Π_{ideal}. □

Note that $\Pr[i = \mathcal{A}_{paths} \mid b = i] \geq \frac{1}{2}$ is always true, since our adversary always guesses unless it is sure to win. So, if exactly one of S_0 and S_1 is non-empty, $\mathcal{A}_{paths}$ certainly wins, otherwise it wins with probability $\frac{1}{2}$. The above fact, along with Claim 1 and Claim 3, helps us conclude that the best chance for any protocol against $\mathcal{A}_{paths}$ is bounded by the probability of Π_{ideal} satisfying Invariant 1.

3.2 Synchronized Users with Non-compromising Adversaries

Our first scenario is a protocol-friendly user distribution U_B, where inputs from all users are globally synchronized: over the course of N rounds, exactly one user per round sends a message, following a random permutation that assigns one round to each user. Analogously, the protocol globally instructs the users to send up to $\beta \in [0, 1]$ noise messages **per user** per round, or $B = \beta\mathsf{N}$ noise messages per round in total.

In real life, the user distribution is independent of the protocol. However, to make the user distribution protocol-friendly in our modeling we consider a globally controlled user distribution. For this scenario, we consider *non-compromising* passive adversaries that can observe all network traffic.

3.2.1 Lower Bound on Adversarial Advantage

Theorem 3.2.1. *For user distribution U_B, no protocol $\Pi \in M$ can provide δ-sender anonymity, for any $\delta < 1 - f_\beta(\ell)$, where $f_\beta(x) = \min(1, ((x + \beta\mathsf{N}x)/(\mathsf{N} - 1)))$.*

Proof. By Claim 3, we know that Π_{ideal} is an optimal protocol for satisfying Invariant 1 and by Claim 1 we know that satisfying Invariant 1 is necessary for anonymity, as otherwise our adversary $\mathcal{A}_{paths}$ can win against the protocol. Thus, the probability that Π_{ideal} satisfies Invariant 1 directly provides a lower bound of the adversary's advantage against any protocol.

Let, u_0 and u_1 be the users chosen by the adversary and let b be the challenge bit. Let t_0 be the round in which u_b sends the challenge message and let r be the round in which the challenge message reaches the recipient.

Recall that Invariant 1 is necessary for the protocol to provide anonymity; u_{1-b} sends her messages (can be a noise message) at $V = \{t_1, t_2, t_3, \ldots, t_k\}$, then $T = \{t : t \in V \wedge (r - \ell) \leq t < r\}$. Since we are considering a non-compromising adversary, $\Pr[\text{Invariant 1 is true}] = \Pr[\text{T is not empty}]$. With the above in mind, let us define the following events:

H_1: In ℓ rounds u_{1-b} sends at least one noise message.

H_2: u_{1-b} sends his own message within the chosen ℓ rounds.

H_3: there is at least one message from u_{1-b} within the chosen ℓ rounds $\equiv T$ is not empty $\equiv$ Invariant 1 is true.

Consider any slice of ℓ rounds around the challenge message, there are exactly $(\ell - 1)$ user messages other than the challenge message. Hence, any slice of ℓ rounds yields the same probability of containing a user message from u_{1-b}, except when $r < \ell$ OR $r > \mathsf{N}$ where the probability is smaller. Thus, no matter what value of t is returned by O, $\Pr[H_2] \leq \frac{\ell-1}{\mathsf{N}-1}$.

Given any values $\ell, \beta \geq 0$, $\mathcal{A}_{paths}$ has the least chance of winning, if for a given interval of ℓ rounds, $\beta\mathsf{N}\ell$ unique users are picked to send the noise messages in such a way that they are not scheduled to send their own messages in that interval. Since, Π_{ideal} needs only one message from u_{1-b} in the interval of ℓ rounds for Invariant 1 to hold, it tries to maximize the number of users that send messages in that interval. Hence, $\Pr[H_1] \leq \frac{\beta\mathsf{N}\ell}{\mathsf{N}-1}$. Therefore,

$$\Pr[\neg H_3] = \Pr[\neg H_1, \neg H_2] \geq \max(0, (\mathsf{N} - \ell - \beta\mathsf{N}\ell)/(\mathsf{N} - 1)).$$
$$\Pr[H_3] = 1 - \Pr[\neg H_3] \leq \min(1, ((\ell + \beta\mathsf{N}\ell)/(\mathsf{N} - 1))).$$

Thus, we can bound the probability for the adversary as
$Pr[0 = \mathcal{A}_{paths}|b = 1] = Pr[1 = \mathcal{A}_{paths}|b = 0] \leq \frac{1}{2}\Pr[H_3]$;
and $Pr[0 = \mathcal{A}_{paths}|b = 0] \geq 1 - \frac{1}{2}\Pr[H_3]$.
And therefore, since $\delta \geq Pr[0 = \mathcal{A}_{paths}|b = 0] - Pr[0 = \mathcal{A}_{paths}|b = 1]$,

$$\delta \geq 1 - \Pr[H_3] \geq 1 - f_\beta(\ell).$$

□

3.2.2 Impossibility for Strong Anonymity

We now investigate under which constraints for ℓ and β Theorem 3.2.1 rules out strong anonymity.

Theorem 3.2.2. *For user distribution U_B with $\ell < \mathsf{N}$ and $\beta\mathsf{N} \geq 1$, no protocol $\Pi \in M$ can achieve strong anonymity if $2\ell\beta < 1 - \epsilon(\eta)$, where $\epsilon(\eta) = \frac{1}{\eta^d}$ for a positive constant d.*

Proof. For strong anonymity, we require: $\delta(\eta) = neg(\eta)$, and we know that for Π_{ideal} we have: $\delta(\eta) \geq 1 - f_\beta(\ell) = \left(\frac{\mathsf{N}-\ell-\beta\mathsf{N}\ell}{\mathsf{N}-1}\right) \geq \left(\frac{\mathsf{N}-\ell-\beta\mathsf{N}\ell}{\mathsf{N}}\right) \geq 1 - \frac{\ell}{\mathsf{N}} - \beta\ell$.
We assume for contradiction that there is a protocol limited by ℓ and β such that $2\ell\beta < 1-\epsilon(\eta)$ that still achieves strong anonymity. Since $\delta(\eta) = neg(\eta)$, we know that $\epsilon(\eta) > \delta(\eta)$.

$$\begin{aligned}
&\epsilon(\eta) > \delta(\eta) \implies \epsilon(\eta) > 1 - \frac{\ell}{\mathsf{N}} - \beta\ell \\
\implies &\epsilon(\eta) > 1 - \frac{\ell}{\mathsf{N}} - \frac{1}{2}\left(1 - \epsilon(\eta)\right) \\
\iff &2\ell > \mathsf{N}\left(1 - \epsilon(\eta)\right) \overset{N\beta\geq 1}{\implies} 2\ell\beta > 1 - \epsilon(\eta)
\end{aligned}$$

The above contradicts the assumption that $2\ell\beta < 1 - \epsilon(\eta)$.

Note: In case $\beta\mathsf{N} < 1$, no noise messages are allowed per round (i.e., $\beta = 0$) and thus $\delta(\eta) \geq 1 - \ell/\mathsf{N}$, which is not negligible unless $\ell = \mathsf{N}$, since $\mathsf{N} = poly(\eta)$. □

Note that this is a necessary constraint for anonymity, but not a sufficient condition. There can exist ℓ and β such that $2\ell\beta > 1 - neg(\eta)$, but Π_{ideal} can not achieve strong anonymity, and hence no protocol can achieve strong anonymity. We have discussed few such examples later in the following part of this section.

Interesting Cases. For illustration, we now discuss a few examples for different values of ℓ, β, and N.

1. If $\ell = \mathsf{N}$, we can have $\delta = 0$ even for $\beta = 0$. Anonymity can be achieved trivially by accumulating all messages from all N users and delivering them together at round $(\mathsf{N}+1)$. In this case $2\ell\beta = 0 < 1 - \epsilon(\eta)$, but also $\beta\mathsf{N} = 0 < 1$.

2. $\beta = \frac{1}{\eta}$, $\ell = \eta$: We have $\delta \geq \frac{\mathsf{N}-\eta-\mathsf{N}}{\mathsf{N}} \geq \frac{-\eta}{\mathsf{N}}$. In ℓ rounds the protocol can send $\ell\beta\mathsf{N} = \mathsf{N}$ noise messages and achieve strong anonymity (all N users send a noise message each).

3. $\beta = \frac{1}{2\eta}$, $\ell = \eta$: Here we have, $\delta \geq \frac{\mathsf{N}-\eta-\frac{\mathsf{N}}{2}}{\mathsf{N}} = \frac{1}{2} - \frac{\eta}{\mathsf{N}}$. In this case, strong anonymity is possible if $\frac{\eta}{\mathsf{N}} \geq \frac{1}{2} - neg(\eta)$. Even though $2\ell\beta = 1 > 1 - neg(\eta)$, anonymity depends on the relation between η and N.

4. $\beta = \frac{1}{2}, \ell = 1$: We have $\delta \geq \frac{\mathsf{N}-1-\frac{\mathsf{N}}{2}}{\mathsf{N}} \approxeq \frac{1}{2}$. In a best scenario, only half of the users send messages in ℓ rounds. Therefore, protocols cannot achieve strong anonymity here even though $2\ell\beta > 1 - neg(\eta)$.

5. $\beta = \frac{1}{9}$, $\ell = 3$: For $\eta > 3$ and $\mathsf{N} > 4$, which is a very natural assumption, we have $2\ell\beta = \frac{2}{3} < 1 - neg(\eta)$. Then, $\delta \geq \frac{\mathsf{N}-3-\frac{N}{3}}{\mathsf{N}} > neg(\eta)$. Then, $\delta \geq \frac{\mathsf{N}-3-\frac{N}{3}}{\mathsf{N}} = 1 - \frac{3}{\mathsf{N}} - \frac{1}{3}$. Here, δ can not be $neg(\eta)$. If we consider our Π_{ideal}, in ℓ rounds it receives only $(\frac{\mathsf{N}}{3} + 3)$ messages (noise + user messages). So a maximum of $(\frac{\mathsf{N}}{3} + 2)$ users can send messages other than the challenge user, and there is a high probability that u_{1-b} has not sent a message. Hence Π_{ideal} cannot achieve strong anonymity, and analogously no other protocol in M can achieve that.

3.3 Synchronized Users with Partially Compromising Adversaries

We now extend our analysis of the previous section by having compromised protocol parties. Given the set of protocol parties P, now our adversary $\mathcal{A}_{paths}$ can compromise a set of c parties $\mathsf{P_c} \subset \mathsf{P}$. If $\mathcal{A}_{paths}$ can compromise all the parties in P, anonymity is broken trivially - that's why we do not analyze that case separately. Recall from Section 3.1.4 that $\mathcal{A}_{paths}$ picks the c parties from P uniformly at random. We consider the same user distribution U_B as in Section 3.2.

3.3.1 Lower Bound on Adversarial Advantage

In our protocol Π_{ideal} the oracle O decides on the time t to deliver each message, which is within $[1, \ell]$, s.t. t maximizes the probability that Invariant 1 is true. Similar to Section 3.2, we now calculate a bound on the probability that Π_{ideal} satisfies Invariant 1.

Theorem 3.3.1. *For user distribution U_B, no protocol $\Pi \in M$ can provide δ-sender anonymity, for any*

$$\delta < \begin{cases} 1 - [1 - \binom{\mathsf{c}}{\ell} / \binom{\mathsf{K}}{\ell}] f_\beta(\ell) & \mathsf{c} \geq \ell \\ 1 - [1 - 1/\binom{\mathsf{K}}{\mathsf{c}}] f_\beta(\mathsf{c}) - f_\beta(\ell - \mathsf{c}) & \mathsf{c} < \ell \end{cases}$$

where $f_\beta(x) = \min(1, ((x + \beta\mathsf{N}x)/(\mathsf{N} - 1)))$.

Proof. Let u_0, u_1 be the challenge users and let b be the challenge bit. Moreover, let t_0 be the time the challenge message is sent by u_b and let $r = t_0 + t$ be the time it is received by the recipient, where t is the delivery time decided by the oracle O.

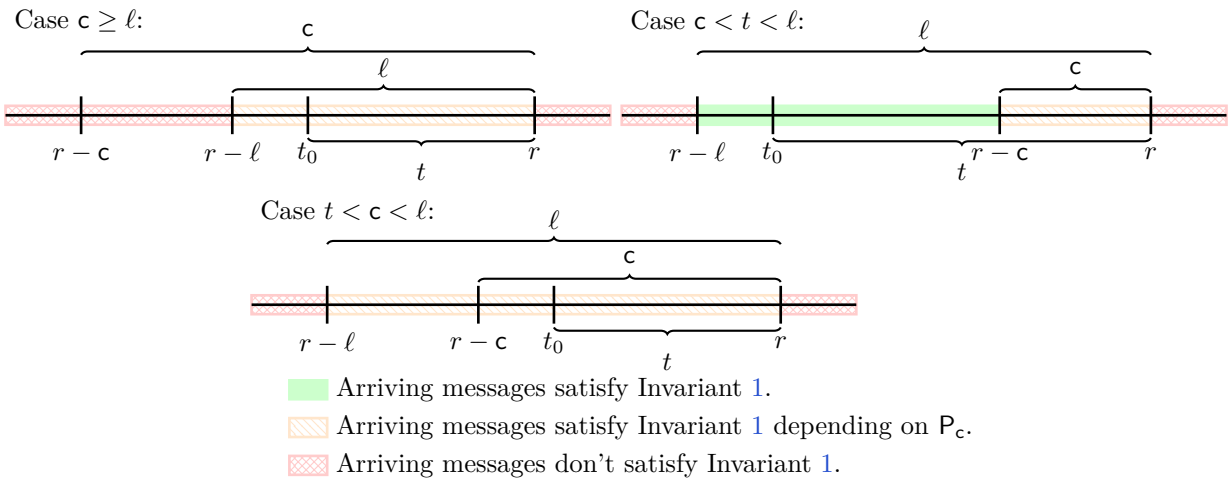


Figure 3.5. Satisfying Invariant 1 depending on the arrival time of messages from u_{1-b} in the cases of the proof for Theorem 3.3.1.

We distinguish two cases, depending on ℓ and c: 1. First, where the number of compromised parties c is at least as large as the maximal latency ℓ. In this case, all parties on the path of the challenge message could be compromised. 2. Second, where not all parties on the path of the challenge message can be compromised. And hence, the analysis focuses on the arrival times of messages from u_{1-b}. For a graphical depiction of the relationship between the rounds a message from u_{1-b} arrives and it satisfying Invariant 1 we refer to Figure 3.5.

1) Case $\mathsf{c} \geq \ell$. We know, $\ell \geq t$ holds by definition. The invariant is true only if u_{1-b} sends at least one message in one of the rounds between $(r - \ell)$ and $(r - 1)$. Additionally, if u_{1-b} sends at least one message in $\{r - \ell, \ldots, t_0\}$, the invariant holds only if there is at least one non-compromised party on the path between t_0 and $(r - 1)$. Whereas, if u_{1-b} does not send any message in $\{r - \ell, \ldots, t_0\}$, and the first message from u_{1-b} in the interval $\{t_0 + 1, r - 1\}$ arrives at t_1, the invariant holds only if there is at least one non-compromised party on the path between t_1 and $(r - 1)$.

Note that $\mathsf{K} > \mathsf{c} \geq \ell$. Also recall from Section 3.1 that $\mathcal{A}_{paths}$ picks the c parties uniformly at random from K parties.

$\Pr[\text{Invariant 1 is true}]$

$\leq \sum_{j=r-\ell}^{t_0} \Pr[u_{1-b}.\text{sent}(j) \wedge \neg u_{1-b}.\text{sent}(j+1, t_0)] \cdot \Pr[\neg\text{Cmpr}(t)]$

$+ \sum_{j=t_0+1}^{r} \Pr[u_{1-b}.\text{sent}(j) \wedge \neg u_{1-b}.\text{sent}(r-\ell, j-1)] \cdot \Pr[\neg\text{Cmpr}(r-j)]$

$\leq \Pr[\neg\text{Cmpr}(\ell)] \cdot \Pr[u_{1-b}.\text{sent}(r-\ell, r-1)]$

$\leq [1 - \binom{\mathsf{c}}{\ell}/\binom{\mathsf{K}}{\ell}] \cdot \min(1, ((\ell + \beta\mathsf{N}\ell)/(\mathsf{N}-1)))$.

By Claim 1 the adversary wins whenever Invariant 1 is not true. Hence, we know that the probability that the adversary guesses incorrectly is bounded by:

$\Pr[0 = \mathcal{A}_{paths} | b = 1] = \Pr[1 = \mathcal{A}_{paths} | b = 0] \leq \frac{1}{2}\Pr[\text{Invariant 1 is true}]$

$\leq \frac{1}{2}[1 - \binom{\mathsf{c}}{\ell}/\binom{\mathsf{K}}{\ell}] \cdot \min(1, (\frac{\ell+\beta\mathsf{N}\ell}{\mathsf{N}-1}))$.

Thus, $\delta \geq 1 - [1 - \binom{\mathsf{c}}{\ell}/\binom{\mathsf{K}}{\ell}] \cdot \min(1, (\frac{\ell+\beta\mathsf{N}\ell}{\mathsf{N}-1}))$.

2) Case $\mathsf{c} \leq \ell$: The probability that all parties on the mutual path of the challenge message and a message from the alternative sender u_{1-b} are compromised now mainly depends on the arrival time of the messages from u_{1-b}. We find two sub-cases depending on the oracle's choice for t.

2a) Case $\mathsf{c} \leq t$:

$\Pr[\text{Invariant 1 is true}]$

$\leq \Pr[u_{1-b}.\text{sent}(r-\ell, r-\mathsf{c})] + \Pr[\neg u_{1-b}.\text{sent}(r-\ell, r-\mathsf{c})]$

$\cdot \Pr[u_{1-b}.\text{sent}(r-\mathsf{c}, r)] \cdot \Pr[\neg\text{Cmpr}(\mathsf{c})]$

$\leq \min(1, (\frac{(\ell-\mathsf{c})+\beta\mathsf{N}(\ell-\mathsf{c})}{\mathsf{N}-1})) + \min(1, (\frac{\mathsf{N}-(\ell-\mathsf{c})-\beta\mathsf{N}(\ell-\mathsf{c})}{\mathsf{N}-1})(\frac{\mathsf{c}+\beta\mathsf{N}\mathsf{c}}{\mathsf{N}-(\ell-\mathsf{c})-\beta\mathsf{N}(\ell-\mathsf{c})}))[1 - \frac{1}{\binom{\mathsf{K}}{\mathsf{c}}}]$

$\leq f_\beta(\ell - \mathsf{c}) + f_\beta(\mathsf{c})[1 - 1/\binom{\mathsf{K}}{\mathsf{c}}]$.

Note that the probability that there are no messages from u_{1-b} in $[(r-\ell), (r-\mathsf{c})]$ and that there is at least one message from u_{1-b} in $[(r-\mathsf{c}), r]$ are not independent from each other. The best thing a protocol can do with the noise messages is to have $\mathsf{N}\beta\ell$ unique users, different from the ℓ users who send their actual message, send the noise messages. Thus, if a user sends a message in $[(r-\ell), (r-\mathsf{c})]$, he can not send a message in $[(r-\mathsf{c}), r]$. The above calculations are done considering that best scenario. Also note that the value of K may be

larger or smaller than ℓ and t, but as long as $\mathsf{c} \leq \mathsf{K}$, the bound given above holds. Hence, $\delta \geq 1 - f_\beta(\ell - \mathsf{c}) - [1 - 1/\binom{\mathsf{K}}{\mathsf{c}}] \cdot f_\beta(\mathsf{c})$.

2b) Case $t < \mathsf{c}$:

$$
\begin{aligned}
&\Pr\left[\text{Invariant 1 is true}\right] \\
&\leq \Pr\left[u_{1-b}.\text{sent}(r-\ell, r-\mathsf{c})\right] \cdot \Pr\left[\neg\text{Cmpr}(t)\right] \\
&\quad + \Pr\left[\neg u_{1-b}.\text{sent}(r-\ell, r-\mathsf{c})\right] \\
&\qquad \cdot \Pr\left[u_{1-b}.\text{sent}(r-\mathsf{c}), r)\right] \cdot \Pr\left[\neg\text{Cmpr}(t)\right] \\
&\leq \Pr\left[u_{1-b}.\text{sent}(r-\ell, r-\mathsf{c})\right] + \Pr\left[\neg u_{1-b}.\text{sent}(r-\ell, r-\mathsf{c})\right] \\
&\qquad\qquad \cdot \Pr\left[u_{1-b}.\text{sent}(r-\mathsf{c}, r)\right] \cdot \Pr\left[\neg\text{Cmpr}(t)\right]
\end{aligned}
$$

The event expression above is the same as in the previous case ($t > \mathsf{c}$). The bound on δ thus follows analogously. □

3.3.2 Impossibility for Strong Anonymity

Theorem 3.3.2. *For user distribution U_B with $\mathsf{K} \in poly(\eta)$, $\mathsf{K} > \mathsf{c} \geq \ell$, $\ell < \mathsf{N}$ and $\beta\mathsf{N} \geq 1$, no protocol $\Pi \in M$ can achieve strong anonymity if $2\ell\beta < 1 - \epsilon(\eta)$ or $\ell \in \mathcal{O}(1)$, where $\epsilon(\eta) = 1/\eta^d$ for a positive constant d.*

Proof. When $\mathsf{c} \geq \ell$: $\delta \geq 1 - \left[1 - \frac{\binom{\mathsf{c}}{\ell}}{\binom{\mathsf{K}}{\ell}}\right] f_\beta(\ell)$.

For δ to become $neg(\eta)$, we need both $[1 - \binom{\mathsf{c}}{\ell}/\binom{\mathsf{K}}{\ell}]$ and $f_\beta(\ell)$ to become overwhelming. From Theorem 3.2.2 and Theorem 3.2.1, we know that $2\ell\beta > 1 - neg(\eta)$ is a necessary condition for $f_\beta(\ell)$ to become overwhelming. Now, we are left with the factor $[1 - \binom{\mathsf{c}}{\ell}/\binom{\mathsf{K}}{\ell}]$.

This can become overwhelming iff $[\binom{\mathsf{c}}{\ell}/\binom{\mathsf{K}}{\ell}]$ becomes negligible. We know that $\mathsf{K} > \mathsf{c} \geq \ell$ and $\mathsf{K} \in poly(\eta)$. Hence, for some constant x,

$$\frac{\mathsf{c}-\ell}{\mathsf{K}-\ell} > \frac{1}{\eta^x} \iff \left(\frac{\mathsf{c}-\ell}{\mathsf{K}-\ell}\right)^\ell > \left(\frac{1}{\eta^x}\right)^\ell$$
$$\implies \frac{\mathsf{c}(\mathsf{c}-1)\ldots(\mathsf{c}-\ell)}{\mathsf{K}(\mathsf{K}-1)\ldots(\mathsf{K}-\ell)} > \left(\frac{\mathsf{c}-\ell}{\mathsf{K}-\ell}\right)^\ell > \left(\frac{1}{\eta^x}\right)^\ell$$
$$\iff \frac{\binom{\mathsf{c}}{\ell}}{\binom{\mathsf{K}}{\ell}} > \left(\frac{1}{\eta^x}\right)^\ell.$$

For any $\ell \in \mathcal{O}(1)$, $(1/\eta^x)^\ell$ is non-negligible. □

To achieve strong anonymity against $\mathcal{A}_{paths}$, we need $\ell \in \omega(1)$, additional to the constraint of $2\ell\beta > 1 - neg(\eta)$. We now focus on the constraint $\ell \in \omega(1)$ and refer to Section 3.2.2 for a comprehensive case study on the other constraint.

Interesting Cases. Now we are going to discuss a few interesting cases for different values of $\ell < \mathsf{c}$, and K.

1. $\ell = \eta$ and $\mathsf{K}/\mathsf{c} = constant$: In this case we have, $\binom{\mathsf{c}}{\ell}/\binom{\mathsf{K}}{\ell} = \frac{\mathsf{c}(\mathsf{c}-1)\ldots(\mathsf{c}-\ell+1)}{\mathsf{K}(\mathsf{K}-1)\ldots(\mathsf{K}-\ell+1)} < (\mathsf{c}/\mathsf{K})^\ell = (\mathsf{c}/\mathsf{K})^\eta$. Hence, $\binom{\mathsf{c}}{\ell}/\binom{\mathsf{K}}{\ell}$ becomes negligible and strong anonymity is possible. Even though c has a high value, because of the high value of ℓ there is a significant possibility that the challenge message will meet a message from u_{1-b} at some honest node, given a high value of β such that $2\ell\beta > 1 - neg(\eta)$.

2. $\ell = \mathcal{O}(1), \mathsf{c} = \mathcal{O}(1)$: Now we have, $\binom{\mathsf{c}}{\ell}/\binom{\mathsf{K}}{\ell} = \frac{\mathsf{c}(\mathsf{c}-1)\ldots(\mathsf{c}-\ell+1)}{\mathsf{K}(\mathsf{K}-1)\ldots(\mathsf{K}-\ell+1)} > ((\mathsf{c}-\ell)/(\mathsf{K}-\ell))^\ell$. But $\mathsf{K} \in poly(\eta)$, and c and ℓ can only have integer values. Hence $((\mathsf{c}-\ell)/(\mathsf{K}-\ell))^\ell$ is non-negligible, and hence $\binom{\mathsf{c}}{\ell}/\binom{\mathsf{K}}{\ell}$ is also non-negligible. Even though c has a small value, ℓ is also small. Hence, it is unlikely that the challenge message will mix with a message from u_{1-b} at some honest node. Thus, strong anonymity cannot be achieved.

3. $\ell = \mathsf{c} = \mathsf{K} - 1$: Now we have, $\binom{\mathsf{c}}{\ell}/\binom{\mathsf{K}}{\ell} = \frac{1}{\mathsf{K}}$. But $\mathsf{K} \in poly(\eta)$. Hence $((\mathsf{c}-\ell)/(\mathsf{K}-\ell))^\ell$ is non-negligible. Therefore, $\binom{\mathsf{c}}{\ell}/\binom{\mathsf{K}}{\ell}$ is non-negligible, no matter what value of β we pick. Even though ℓ might have a high value (depending on the value of K), c is equally high. Hence, it is unlikely that the challenge message will mix with a message from u_{1-b} at some honest node. Thus, strong anonymity cannot be achieved, despite the necessary constraints being

satisfied. This reflects the fact that the constraints are necessary for anonymity, but not sufficient conditions for anonymity.

Theorem 3.3.3. *For user distribution U_B with $\mathsf{K} \in poly(\eta)$, $\mathsf{c} \in \mathcal{O}(1)$, $\mathsf{K} > \ell > \mathsf{c}$, $\ell < \mathsf{N}$ and $\beta\mathsf{N} \geq 1$, no protocol $\Pi \in M$ can achieve strong anonymity if $2(\ell-\mathsf{c})\beta < 1-\epsilon(\eta)$, where $\epsilon(\eta) = \frac{1}{\eta^d}$ for a positive constant d.*

Proof. When $\mathsf{c} < \ell$: $\delta \geq 1 - \left[1 - 1/\binom{\mathsf{K}}{\mathsf{c}}\right] f_\beta(\mathsf{c}) - f_\beta(\ell - \mathsf{c})$.

First consider the factor $[1 - 1/\binom{\mathsf{K}}{\mathsf{c}}]$. Since $\mathsf{K} = poly(\eta)$ and $\mathsf{c} = constant$, $[1/\binom{\mathsf{K}}{\mathsf{c}}]$ can never be negligible. And thus, $[1 - 1/\binom{\mathsf{K}}{\mathsf{c}}]$ can never be overwhelming. So, $[1 - 1/\binom{\mathsf{K}}{\mathsf{c}}]f_\beta(\mathsf{c})$ can never be overwhelming as well, since $f_\beta(\mathsf{c}) \leq 1$.

Now, let's consider $f_\beta(\ell - \mathsf{c})$ and $f_\beta(\mathsf{c})$. Note that, these two factors represent the probabilities of two dependent but mutually exclusive events, and hence $f_\beta(\mathsf{c}) + f_\beta(\ell - \mathsf{c}) \leq 1$. And we already know that $[1 - 1/\binom{\mathsf{K}}{\mathsf{c}}]$ can never be overwhelming. Thus, the only way δ can become negligible is if $f_\beta(\ell - \mathsf{c})$ becomes overwhelming. Note that, if $a + b \leq 1$ and $c < 1$, the only way $ac + b = 1$ is possible if $b = 1$.

Now we can follow exactly the same procedure as in the proof of Theorem 3.2.2 to say: $f_\beta(\ell - \mathsf{c})$ can not become overwhelming if $2(\ell - \mathsf{c})\beta < 1 - \epsilon(\eta)$. □

The analysis in this case is exactly same as Section 3.2.2, except that here we need to consider the slice of $(\ell - \mathsf{c})$ rounds instead of ℓ rounds.

It is worth repeating here, all the constraints we have derived in Section 3.2 and Section 3.3 are necessary for anonymity, but they are not sufficient conditions for anonymity.

3.4 Unsynchronized Users with Non-compromising Adversaries

In this and the subsequent section we use an unsynchronised user distribution U_P: In each round, independent of other users and other rounds, each client tosses a biased coin with success probability $p \in (0, 1]$. On a success the client sends a message in that round, otherwise it does not send a message. Consequently, the number of messages per round follows Binomial distribution $Binom(\mathsf{N}, p)$ if the number of users N is large and p sufficiently small, the resulting binomial distribution reduces to a Poisson distribution, which is a close approximation of real-life traffic patterns.

For a protocol with bandwidth overhead β, we distinguish between the actual probability that users want to send messages p and the value for p that we use in our analysis, i.e., we set $p = p + \beta$. In this unsynchronised scenario the bandwidth of genuine messages contributes to the anonymity bound.

As in Section 3.2, in this section we consider a *non-compromising* adversary.

3.4.1 Lower Bound on Adversarial Advantage

Theorem 3.4.1. *For user distribution U_P, no protocol $\Pi \in M$ can provide δ-sender anonymity, for any $\delta < 1 - \left(\frac{1}{2} + f_p(\ell)\right)$, where $f_p(x) = \min(1/2,\ 1-(1-p)^x)$ for a positive integer x.*

Proof. Similar to Section 3.2, we calculate a bound on the probability that Π_{ideal} satisfies Invariant 1, and that bound is valid against any other protocol in our model. Since we consider a non-compromising adversary, $\Pr[\text{Invariant 1 is True}] = \Pr[\text{T is not empty}]$, where T is defined as in Invariant 1.

Let us consider the random variables $X^{(1)}, X^{(2)}, \ldots, X^{(\mathsf{N})}$, where $X^{(i)}$ denotes the event of the i^{th} user sending her own message within a given interval of ℓ rounds $[a, b]$, with $(b-a) = \ell$. All $X^{(i)}$s are mutually independent and we have,

$$X^{(i)} = \begin{cases} 0 & \text{with probability } (1-p)^{\ell} \\ 1 & \text{with probability } (1-(1-p)^{\ell}). \end{cases}$$

Next, let $X = \sum_{i=1}^{\mathsf{N}} X^{(i)}$ be a random variable representing the number of users that send messages in an interval of ℓ rounds. We calculate for the expected value $\mathbb{E}[X]$ of X,

$$\begin{aligned} \mathbb{E}[X] &= \mathbb{E}[\textstyle\sum_{i=1}^{\mathsf{N}} X^{(i)}] = \sum_{i=1}^{\mathsf{N}} \mathbb{E}[X^{(i)}] \\ &= \mathsf{N}(1-(1-p)^{\ell}) = \mu. \end{aligned}$$

Using the Chernoff Bound on the random variable X we derive $\Pr[X - \mu \geq \mathsf{N}a] \leq \exp(-2a^2\mathsf{N})$, which for $a = \frac{\mu}{\mathsf{N}}$ lets us estimate, $\Pr[X \geq 2\mu] \leq \exp(-2(\mu^2/\mathsf{N}^2)\mathsf{N})$. For brevity in the following calculation we denote, $Pr[X \geq 2\mu]$ by E and the event that T is non-empty by Y and since all users are acting independently from each other we get for $j \in \{0, \ldots, \mathsf{N}\}$, $\Pr[Y|X = j] = 1 - \Pr[\neg Y|X = j] = \frac{j}{\mathsf{N}}$.

For $2\mu \leq \mathsf{N}$, we have the following,

$$\begin{aligned}
&\Pr[Y] \\
=&\Pr[X \geq 2\mu] \cdot \Pr[Y|X \geq 2\mu] + \Pr[X < 2\mu] \cdot \Pr[Y|X < 2\mu] \\
\leq&\Pr[X \geq 2\mu] \cdot \Pr[Y|X = \mathsf{N}] + \Pr[X < 2\mu] \cdot \Pr[Y|X = 2\mu] \\
=&E \cdot \Pr[Y|X = \mathsf{N}] + (1-E) \cdot \Pr[Y|X = 2\mu] \\
=&E \cdot \frac{\mathsf{N}}{\mathsf{N}} + (1-E) \cdot \frac{2\mu}{\mathsf{N}} \quad = 1 - (1-E)(1-2f_p(\ell)).
\end{aligned}$$

If $2\mu > \mathsf{N}$, we get with $f(\ell) = \min\left(\frac{1}{2},\ 1-(1-p)^{\ell}\right)$, $\Pr[Y] \leq E + (1-E)1 \leq 1 \leq 1 - (1-E)(1-2f_p(\ell))$.

Thus, $\delta \geq 1 - \Pr[Y] \geq (1-E)(1-2f_p(\ell))$. We now use Markov's Inequality on X and derive $E = \Pr[X \geq 2\mu] \leq \frac{1}{2}$, which means, $\delta \geq \frac{1}{2}(1-2f_p(\ell)) \geq \frac{1}{2} - f_p(\ell)$. □

Note that in the proof of Theorem 3.4.1, in case p is a constant and N is a very high value, then E goes towards zero and instead of using Markov's inequality, we can derive $\delta \geq 1 - 2f_p(\ell)$.

3.4.2 Impossibility for Strong Anonymity

Theorem 3.4.2. *For user distribution U_P and $p > 0$, no protocol $\Pi \in M$ can achieve strong anonymity if $2\ell p < 1 - \epsilon(\eta)$, where $\epsilon(\eta) = 1/\eta^d$ for a positive constant d.*

Proof. We know $0 \leq E \leq 1/2$. When $2\mu \leq \mathsf{N}$,

$$\begin{aligned}
\delta \geq&(1-E)(1-2f_p(\ell)) \geq 1/2\left(2(1-p)^{\ell} - 1\right) \\
\geq&1/2\left(2(1-\ell p) - 1\right) = 1/2(1-2\ell p).
\end{aligned}$$

Thus, if $2\ell p < 1 - \epsilon(\eta)$,

$$\begin{aligned}
&2\ell p < 1 - \epsilon(\eta) \iff 1 - 2\ell p > \epsilon(\eta) \\
&\implies \delta > 1/2 \cdot \epsilon(\eta) = \text{non-negligible}.
\end{aligned}$$

Thus, when $2\mu \leq \mathsf{N}$, a necessary condition for δ to become negligible is $2\ell p > 1 - neg(\eta)$.

When $2\mu > \mathsf{N}$, using $\mu = \mathsf{N}(1-(1-p)^{\ell})$ we get:

$$2\mathsf{N}(1-(1-p)^{\ell}) > \mathsf{N} \implies (1-p)^{\ell} < 1/2$$
$$\implies 1 - p\ell < 1/2 \iff 2p\ell > 1.$$

□

Similar to the constraints in Section 3.2 and Section 3.3, this is also a necessary constraint for anonymity, not a sufficient condition. There can exist ℓ and p such that $2\ell p > 1 - neg(\eta)$, but still no protocol can achieve strong anonymity.

Interesting Cases. Now we are going to discuss a few interesting cases for different values of ℓ, p, and N.

1. $p = \frac{1}{\eta}$, $\ell = \eta$: Here, $f_p(\ell) = 1-(1-p)^{\ell} > 1 - 1/e > \frac{1}{2}$. Hence, $\delta \geq \frac{1}{2} - f_p(\ell) = 0$. Since $p\ell = 1$, in ℓ rounds the protocol has 1 message per user on an average. So, the protocol has a high chance of winning, but depending on the specific instance of the user distribution. Whereas in Section 3.2.2, we saw that, for a similar bandwidth and latency overhead, protocols could win with all instances of the synchronized user distribution.

2. $p = \frac{1}{2\eta}$, $\ell = \eta$: even for $\eta > 2$, $f_p(\ell) = 1-(1-p)^{\ell} < 0.45$. Hence, $\delta \geq \frac{1}{2} - f_p(\ell) > 0.05$. Even though $2\ell p = 1$, strong anonymity can not be achieved in this case. In an expected scenario, in a slice of ℓ rounds only $p\ell = \frac{1}{2}$ portion of the total users send messages, and hence there is a significant chance that u_{1-b} is in the other half. Note that this is different from the scenario with synchronized users where protocols could achieve strong anonymity in this case (c.f. Section 3.2.2).

3. p $= \frac{1}{2}$, $\ell = 1$: We have, $f_p(\ell) = 1-(1-p)^{\ell} = \frac{1}{2}$. Hence, $\delta \geq 0$. Although we have $2\ell p = 1$, because of low $\ell(=1)$, u_{1-b} does not send a message with high probability$(=\frac{1}{2})$. This case again highlights that the requirement $2\ell p \geq 1 - \epsilon(\eta)$ is not necessarily sufficient: As in Section 3.2.2, protocols can not achieve strong anonymity in such a situation.

4. p $= \frac{1}{9}$, $\ell = 3$: Here, $f_p(\ell) = 1-(1-p)^{\ell} = 1-\left(\frac{8}{9}\right)^3 < 0.29$, and $\delta \geq \frac{1}{2} - f_p(\ell) > 0.21$; because of low values of both p and ℓ only a few users send messages within the interval of ℓ rounds, and hence the protocol has a small chance to win. As in Section 3.2.2, protocols can not achieve strong anonymity in this case, since the necessary constraints are not satisfied.

3.5 Unsynchronized Users with Partially Compromising Adversaries

Finally, we consider partially compromising adversaries that can compromise a set of c parties $\mathsf{P_c} \subset \mathsf{P}$ for the user distribution U_P defined in Section 3.4.

3.5.1 Lower Bound on Adversarial Advantage

Theorem 3.5.1. *For user distribution U_P, no protocol $\Pi \in M$ can provide δ-sender anonymity, for any*

$$\delta < \begin{cases} 1 - [1 - \binom{\mathsf{c}}{\ell}/\binom{\mathsf{K}}{\ell}][\tfrac{1}{2} + f_p(\ell)] & \mathsf{c} \geq \ell \\ \Big(1 - [1 - 1/\binom{\mathsf{K}}{\mathsf{c}}][\tfrac{1}{2} + f_p(\mathsf{c})]\Big) & \\ \quad \cdot \Big(1 - [1/2 + f_p(\ell - \mathsf{c})]\Big) & \mathsf{c} < \ell \end{cases}$$

where $f_p(x) = \min(1/2,\ 1 - (1-p)^x)$ for a positive integer x.

We derive the bound in Section 3.B by combining the techniques presented in Section 3.3 and Section 3.4. Since the proof does not introduce novel techniques, we omit it and instead refer the interested reader to Section 3.B for the proof.

3.5.2 Impossibility for Strong Anonymity

To analyze the negligibility condition of δ in this scenario, we heavily borrow the analyses that we already have conducted in Section 3.4.2 and Section 3.3.2. We are going to analyze this scenario in two parts:

Case $\mathsf{c} \geq \ell$: We have, $\delta \geq 1 - [1 - \binom{\mathsf{c}}{\ell}/\binom{\mathsf{K}}{\ell}] \left[\tfrac{1}{2} + f_p(\ell)\right]$.

To make δ negligible, both the factors $[1 - \binom{\mathsf{c}}{\ell}/\binom{\mathsf{K}}{\ell}]$ and $[1/2 + f_p(\ell)]$ have to become overwhelming. From Theorem 3.3.2, we know that we need $\ell \in \omega(1)$ to make $[1 - \binom{\mathsf{c}}{\ell}/\binom{\mathsf{K}}{\ell}]$ overwhelming. This is a necessary condition, but not sufficient. For a detailed discussion, we refer to Section 3.3.2. From Section 3.4.2 we know that the necessary condition for $[1/2 + f_p(\ell)]$ to be overwhelming is $2\ell p > 1 - neg(\eta)$. Hence, both conditions are necessary to achieve strong anonymity.

Case $\mathsf{c} < \ell$: We have,

$\delta \geq (1 - [1/2 + f_p(\ell - \mathsf{c})])(1 - [1 - 1/\binom{\mathsf{K}}{\mathsf{c}}][1/2 + f_p(\mathsf{c})])$.

In the above expression, we can see two factors:
(i) $F_1 = (1 - [\frac{1}{2} + f_p(\ell - \mathsf{c})])$, (ii) $F_2 = (1 - [1 - 1/\binom{\mathsf{K}}{\mathsf{c}}][\frac{1}{2} + f_p(\mathsf{c})])$.

To make δ negligible, it suffices that F_1 or F_2 become negligible. Unlike Section 3.3, here $f_p(\ell - \mathsf{c})$ and $f_p(\mathsf{c})$ are independent, which allows us to analyze F_1 and F_2 independently. First, F_1 is similar to the δ-bound in Section 3.4, except that we consider $f_p(\ell - \mathsf{c})$ instead of $f_p(\ell)$. Hence, the analysis of F_1 is analogous to Section 3.4.2. Second, F_2 is negligible if both $[1 - 1/\binom{\mathsf{K}}{\mathsf{c}}]$ and $[1/2 + f_p(\mathsf{c})]$ are overwhelming. From Section 3.3.2 we know that $[1 - 1/\binom{\mathsf{K}}{\mathsf{c}}]$ can not be overwhelming for a constant c. Moreover, $f_p(\mathsf{c})$ can be analyzed exactly as $f_p(\ell)$ in Section 3.4.2.

3.6 Recipient Anonymity

The protocol model remains unchanged for recipient anonymity with the exception that the colored token now additionally has a private field for the recipient. We also require noise messages to adhere to the latency bound ℓ. Now we assume that there are N recipients in $\mathcal{R}$. Since, we are not concerned about distinguishing senders, we can assume that there is only one sender in $\mathcal{S}$.

The anonymity game also remains almost same as Section 3.1.2, with only one change: now the game uses α_{RA} [39] instead of α_{SA} as the anonymity notion. Naturally, the adversary is not informed about the delivery of the challenge message by a recipient, but of the sending of the challenge message by a sender.

The adversarial strategy $\mathcal{A}_{paths}$ also remains similar to that of sender anonymity scenario. But here, the adversary can identify the challenge message when it is sent by the sender, not when received by a recipient. After observing the tokens, $\mathcal{A}_{paths}$ tries to construct possible paths for the challenge message to the challenge recipients.

More formally, Given two challenge recipients R_0 and R_1 and the set of observed tokens $(t, r) \in \mathsf{Tokens}$, where t is the token and r the round in which the token was observed, an adversary can construct the sets S_j (for $j \in \{0, 1\}$). Assume the challenge message is sent by the sender u in a round s. We construct possible paths of varying length k, s.t., each element $p \in S_j$ represents a possible path of the challenge message starting from the sender

u in round $r_1 = s$ and the challenge message then arrives at R_j ($j \in \{0,1\}$) in round r_k. With challenge bit b, S_b cannot be empty, as the actual path taken by the challenge message has to be one element in S_b.

$$
\begin{aligned}
S_j = \{p = (t_1.\mathsf{prev}, \ldots, t_k.\mathsf{prev}, t_k.\mathsf{next}) : \\
((t_1, r_1), \ldots, (t_k, r_k)) \in \mathsf{Tokens} \text{ s.t.} \\
t_1.\mathsf{prev} = u \wedge t_k.\mathsf{next} = R_j \\
\wedge\, t_1.\mathsf{msg} = \texttt{Chall} \wedge k \leq \ell \\
\wedge\, \forall_{i \in \{1,\ldots,k-1\}} (t_i.\mathsf{next} = t_{i+1}.\mathsf{prev} \wedge r_{i+1} = r_i + 1 \\
\wedge\, (\ \exists t_{i+1} : (t_{i+1}, r_{i+1}) \in \mathsf{Tokens}\ \wedge t_{i+1}.\mathsf{prev} = t_i.\mathsf{next} \\
\wedge\, t_{i+1}.\mathsf{ID_t} = t_i.\mathsf{ID_t}) \Rightarrow t_{i+1} = t_{i+1})\}
\end{aligned}
$$

Necessary invariant for recipient anonymity. For recipient anonymity it is necessary that at least both challenge recipients receive messages in the ℓ rounds after the challenge message was sent. Moreover, on the path of the actual challenge message, there needs to be at least one honest (non-compromised) party, as otherwise the adversary can track the challenge message from the sender to the recipient (S_b will have exactly one element and S_{1-b} will be empty). Those two conditions together form our *necessary protocol invariant.*

Invariant 2. *Let R_0 and R_1 be the challenge recipients; let b be the challenge bit; and let s be the time when the sender u sends the challenge message towards R_b. Assume that messages for R_{1-b} (including noise messages) are received by R_{1-b} at times $V_{RA} = \{t_1, t_2, t_3, \ldots, t_k\}$. Now, let $T_{RA} = \{t : t \in V_{RA} \wedge s < t \leq (s + \ell)\}$. Then,*

(i) the set T_{RA} is not empty, and

(ii) the challenge message passes through at least one honest node at some time t such that $s \leq t \leq max(T_{RA})$.

The invariant is very similar to Invariant 1 with the only difference that we consider messages sent towards recipients (instead of messages sent by users). In contrast, for sender anonymity, where *sending messages* was the main criteria, for recipient anonymity analo-

gously receiving messages is the main criteria and the times at which messages are received can be (partially) controlled by the protocol.

Claim 4 (Invariant 2 is necessary for anonymity)**.** *Let* Π *be any protocol* $\in M$ *with latency overhead* ℓ *and bandwidth overhead* β*. Let* u, R_0, R_1, b *and* T_{RA} *be defined as in Invariant 2. If Invariant 2 is not satisfied, then our adversary* $\mathcal{A}_{paths}$ *as in Definition 4.3.1 wins (against recipient anonymity).*

Proof. We distinguish two cases, depending on T_{RA}: either T_{RA} is empty, or T_{RA} is non-empty.

If the set T_{RA} is empty, then S_{1-b} is empty as well. However, by construction of our protocol model, the set S_b is always non-empty. Consequently, the adversary $\mathcal{A}_{paths}$ will output b and thus win with probability 1. If T_{RA} is not empty, the following cases can occur:

1. The challenge message never passes through an honest node: In this case, the field $\mathsf{ID_t}$ of the message never changes for the tokens. Thus, S_b will have exactly one element, and S_{1-b} will be an empty set, and consequently $\mathcal{A}_{paths}$ wins.

2. The challenge message passes through one or more honest nodes at times t, such that $t > max(T_{RA})$, but not before: Following the same reasoning as above, we see that paths before $max(T_{RA})$ can be ambiguous, but none of them leads to R_{1-b}. Hence, S_b can have multiple elements, but S_{1-b} will still be an empty set. Thus, $\mathcal{A}_{paths}$ wins.

3. The challenge message passes through an honest node at time t with $t \leq max(T_{RA})$: In this case, the invariant is true.

In all of the above mentioned cases either the invariant is true, or the adversary wins with probability 1. □

Claim 5 (Internally terminated noise does not influence Invariant 2)**.** *Any message that is not delivered to a recipient* $R \in \mathcal{R}$ *does not influence the probability for Invariant 2 being true.*

The proof for this claim is analogous to the proof for Claim 2, where instead of considering the sending of messages, we are concerned with receiving messages.

Ideal Protocol. Now we construct our ideal protocol Π_{ideal}, which is very similar to that of the sender anonymity scenario. The routing strategy is exactly the same as the sender

anonymity scenario. Even for deciding the *optimal* delivery time t for each message, the protocol queries oracle O. The oracle O returns the values in such way that it maximizes the probability of satisfying Invariant 2 for the given routing strategy. The only difference here is that, in sender anonymity scenario the optimal delivery time t for each message is independent of the optimal delivery times for other messages, but here they are dependent.

Claim 6 (Ideal protocol is ideal for the invariant)**.** Π_{ideal} *satisfies Invariant 2 with a probability at least as high as any other protocol in* M*, against the given adversary* $\mathcal{A}_{paths}$*.*

Proof. We want to prove our claim by contradiction. For contradiction, we assume that Π_{ideal} is not the best protocol. That means, there exist a protocol Π_{new} which satisfy Invariant 2 with a higher probability than Π_{ideal}.

We construct a protocol Π_{hybrid} which exactly follows the routing strategy of protocol Π_{ideal}, but for each message Π_{hybrid} selects the time delay same as Π_{new} instead of querying from oracle O.

Let's compare the protocols Π_{hybrid} and Π_{new}. Since both the protocol select the same time delays for every message, the set T_{RA} is exactly same for both of them. Consequently, the path length for the challenge message till $max(T_{RA})$ is same for both the protocols. But, Π_{hybrid} maximizes the number of distinct parties on the path, hence maximizes the probability of having at least one honest party on the path. Therefore, Π_{hybrid} has a probability of satisfying Invariant 2 at least as high as Π_{new}.

Next, let's compare Π_{hybrid} and Π_{ideal}: They both employ the same routing strategy. But Π_{ideal} selects the time delays for all messages by querying oracle O. By definition of the oracle, it selects the time delays in such a way that it maximizes the probability of satisfying Invariant 2, for the given routing strategy. Hence, Π_{ideal} satisfies Invariant 2 with a probability at least as high as Π_{hybrid}. Thus, Π_{new} does not satisfy Invariant 2 with a probability higher than Π_{ideal} - which contradicts our initial assumption that Π_{ideal} is not the best protocol. □

3.6.1 Recipient Anonymity of Synchronized Users with Non-compromising Adversaries

As for our first scenario for sender anonymity, we investigate an ideal user distribution where inputs from all users are globally synchronized.

We assume that all the input messages come within N rounds, exactly one message per round, following a random permutation that assigns one round to each recipient. Formally we group together all users into one sender that sends all the messages. In a given round, the sender should send a message for the assigned recipient. Then, the protocol decides when to deliver the message to the recipient, but not delaying more than ℓ rounds.

We denote this user distribution with U_B. Since, we are considering a globally controlled user distribution, we are considering a globally controlled noise as well. The protocol can add a maximum of $B = \beta\mathsf{N}$ noise messages per round, or β noise messages **per recipient** per round, where $0 \leq \beta \leq 1$. We consider a *non-compromising* passive adversary that can observe all network traffic.

Theorem 3.6.1. *For user distribution U_B, no protocol $\Pi \in M$ can provide δ-recipient anonymity for any $\delta < 1 - f_\beta^{RA}(\ell)$, where $f_\beta^{RA}(d) = \min\left(1, \left(\frac{(\ell+d)+(\ell+d)\beta\mathsf{N}}{\mathsf{N}}\right)\right)$.*

Proof. By Claim 6, we know that Π_{ideal} has the highest probability to satisfy Invariant 2 against $\mathcal{A}_{paths}$. Thus, by Claim 4 it suffices to calculate the probability for Π_{ideal} to satisfy the invariant as a lower bound of the adversary's advantage against any protocol.

Let, R_0 and R_1 be the recipients chosen by the adversary and let b be the challenge bit. Let s be the round in which the sender sends the challenge message.

Recall that Invariant 2 is necessary for the protocol to provide anonymity. Since we are considering a non-compromising adversary, $\Pr[\text{Invariant 2 is true}] = \Pr[\text{T is not empty}]$. If a message is sent for the recipient R_{1-b} (enters the protocol) in $[s-\ell, s+\ell-1]$, it has a possibility to populate an element in T_{RA}. Now let us define the following events:

H_1: Within 2ℓ rounds a noise message is sent to R_{1-b}.

H_2: Within 2ℓ rounds a user sends a real message to R_{1-b}.

H_3: Invariant 2 is true.

We proceed analogously to the proof for Theorem 3.2.1 and get:

$$\Pr[H_2] \leq \frac{2\ell}{\mathsf{N}}.$$

Similarly, in each round noise messages are sent to $\beta\mathsf{N}$ unique users in such a way that no real message is scheduled for them. Thus, $\Pr[H_1] \leq \frac{2\ell\beta\mathsf{N}}{\mathsf{N}}$. We combine these insights to yield a bound.

$$\begin{aligned}\Pr[H_3] &= \Pr[H_1 \vee H_2] \\ &= \min(1, \Pr[H_1 \vee H_2]) \\ &\leq \min(1, \Pr[H_1] + \Pr[H_2]) \\ &\leq \min\left(1, \frac{2\ell + 2\ell\beta\mathsf{N}}{\mathsf{N}}\right).\end{aligned}$$

Thus, we can bound the probability for the adversary as:

$Pr[0 = \mathcal{A}_{paths}|b = 1] = Pr[1 = \mathcal{A}_{paths}|b = 0] \leq \frac{1}{2}\Pr[H_3]$;
and $Pr[0 = \mathcal{A}_{paths}|b = 0] \geq 1 - \frac{1}{2}\Pr[H_3]$.
And therefore, since $\delta \geq Pr[0 = \mathcal{A}_{paths}|b = 0] - Pr[0 = \mathcal{A}_{paths}|b = 1]$, we can say $\delta \geq 1 - \Pr[H_3] \geq 1 - f_{\beta}^{RA}(\ell)$. □

Impossibility for Strong Recipient Anonymity. We now investigate under which constraints for ℓ and β Theorem 3.6.1 rules out strong recipient anonymity.

Theorem 3.6.2. *For user distribution U_B with $\ell < \mathsf{N}$ and $\beta\mathsf{N} \geq 1$, no protocol in M can achieve strong recipient anonymity if $4\ell\beta < 1 - \epsilon(\eta)$, where $\epsilon(\eta) = \frac{1}{\eta^d}$ for a positive constant d and the security parameter η.*

The proof follows analogously to the proof of Theorem 3.2.2.

3.6.2 Recipient Anonymity of Synchronized Users with Partially Compromising Adversaries

Now we extend our analysis for recipient anonymity against partially compromising adversaries, with the same user distribution as the previous section.

Theorem 3.6.3. *No protocol $\Pi \in M$ can provide δ-recipient anonymity for the user distribution U_B, where*

$$\delta < \begin{cases} 1 - \left[1 - \frac{\binom{\mathsf{c}}{\ell}}{\binom{\mathsf{K}}{\ell}}\right] f_\beta^{RA}(\ell) & \mathsf{c} \geq \ell \\ 1 - \left[1 - \frac{1}{\binom{\mathsf{K}}{\mathsf{c}}}\right] f_\beta^{RA}(\mathsf{c}) - f_\beta^{RA}(\ell - \mathsf{c}) & \mathsf{c} < \ell \end{cases}$$

where $f_\beta^{RA}(d) = \min\left(1, \left(\frac{(\ell+d)+(\ell+d)\beta\mathsf{N}}{\mathsf{N}}\right)\right)$.

Proof. Let R_0, R_1 be the challenge users and let b be the challenge bit. Moreover, let s_0 be the time the challenge message is sent for R_b and let $r = s_0 + t$ be the time it is received by the recipient, where t is the delivery time decided by the oracle O for the challenge message.

We distinguish two cases, depending on ℓ and c: 1. First, where the number of compromised parties c is at least as large as the maximal latency ℓ. In this case, all parties on the path of the challenge message could be compromised. 2. Second, where all parties on the path of the challenge message can not be compromised. And hence, the analysis focuses on the delivery times of messages for R_{1-b}.

1) Case $\mathsf{c} \geq \ell$. We know, $\ell \geq t$ holds by definition. The invariant is true if and only if R_{1-b} receives at least one message in one of the rounds between $(s_0 + 1)$ and $(s_0 + \ell)$ and for the last of those messages, delivered at time t_{last}, there is at least one non-compromised party on the path between t_0 and t_{last}. Hence,

$$\begin{aligned} &\Pr\left[\text{Invariant 2 is true}\right] \\ &= \Pr\left[R_{1-b} \text{ receives at least one message in } [s_0, s_0 + \ell]\right] \\ &\qquad \cdot \Pr\left[NOT \text{ all the c parties are compromised}\right] \\ &\leq f_\beta^{RA}(\ell)\left[1 - \frac{\binom{\mathsf{c}}{\ell}}{\binom{\mathsf{K}}{\ell}}\right]. \end{aligned}$$

Hence, $\delta \geq 1 - \left[1 - \frac{\binom{\mathsf{c}}{\ell}}{\binom{\mathsf{K}}{\ell}}\right] f_\beta^{RA}(\ell)$

2) Case $\mathsf{c} \leq \ell$:

The probability that all parties on the mutual path of the challenge message and a message for the alternative recipient R_{1-b} are compromised now mainly depends on the delivery time of the messages for R_{1-b}. We distinguish two sub-cases depending on the oracle's choice for t:

2a) Case $\mathsf{c} \leq t$:

$$
\begin{aligned}
&\Pr\left[\text{Invariant 2 is true}\right] \\
&\leq \Pr\left[R_{1-b} \text{ receives at least one message in } [s_0+\mathsf{c}, s_0+\ell]\right] \\
&\quad + \Pr\left[R_{1-b} \text{ does NOT receive a message in } [s_0+\mathsf{c}, s_0+\ell]\right] \\
&\qquad \cdot \Pr\left[R_{1-b} \text{ receives at least one message in } [s_0, s_0+\mathsf{c}]\right] \\
&\qquad \cdot \Pr\left[NOT \text{ all the c parties are compromised}\right] \\
&\leq f_\beta^{RA}(\ell-\mathsf{c}) + f_\beta^{RA}(\mathsf{c}) \left[1 - \frac{1}{\binom{\mathsf{K}}{\mathsf{c}}}\right].
\end{aligned}
$$

Hence, $\delta \geq 1 - \left[1 - \frac{1}{\binom{\mathsf{K}}{\mathsf{c}}}\right] f_\beta^{RA}(\mathsf{c}) - f_\beta^{RA}(\ell-\mathsf{c})$.

2b) Case $t < \mathsf{c}$:

$$
\begin{aligned}
&\Pr\left[\text{Invariant 2 is true}\right] \\
&\leq \Pr\left[R_{1-b} \text{ receives at least one message in } [s_0+\mathsf{c}, s_0+\ell]\right] \\
&\qquad \cdot \Pr\left[NOT \text{ all the t parties are compromised}\right] \\
&\quad + \Pr\left[R_{1-b} \text{ does NOT receive any message in } [s_0, s_0+\ell]\right] \\
&\qquad \cdot \Pr\left[R_{1-b} \text{ receives at least one message in } [s_0, s_0+\mathsf{c}]\right] \\
&\qquad \cdot \Pr\left[NOT \text{ all the t parties are compromised}\right] \\
\\
&\leq \Pr\left[R_{1-b} \text{ receives at least one message in } [s_0+\mathsf{c}, s_0+\ell]\right] \\
&\quad + \Pr\left[R_{1-b} \text{ does NOT receive any message in } [s_0, s_0+\ell]\right] \\
&\qquad \cdot \Pr\left[R_{1-b} \text{ receives at least one message in } [s_0, s_0+\mathsf{c}]\right] \\
&\qquad \cdot \Pr\left[NOT \text{ all the t parties are compromised}\right]
\end{aligned}
$$

The above event expression is exactly the same as the expression we had in the previous case ($t > \mathsf{c}$). The bound on δ thus follows analogously. □

Impossibility for Strong Recipient Anonymity. We now investigate under which constraints for c, ℓ and β Theorem 3.6.1 rules out strong recipient anonymity.

Theorem 3.6.4. *For user distribution U_B with $\mathsf{K} \in poly(\eta)$, $\mathsf{K} > \mathsf{c} \geq \ell$, $\ell < \mathsf{N}$ AND $\beta\mathsf{N} \geq 1$, no protocol can achieve strong anonymity if $4\ell\beta < 1 - \epsilon(\eta)$ OR $\ell \in \mathcal{O}(1)$, where $\epsilon(\eta) = 1/\eta^d$ for a positive constant d.*

The proof follows analogously to the proof of Theorem 3.3.2.

Theorem 3.6.5. *For user distribution U_B with $\mathsf{K} \in poly(\eta)$, constant c, $\mathsf{K} > \ell > \mathsf{c}$, $\ell < \mathsf{N}$ AND $\beta\mathsf{N} \geq 1$, no protocol can achieve strong anonymity if $4(\ell - \mathsf{c})\beta < 1 - \epsilon(\eta)$, where $\epsilon(\eta) = \frac{1}{\eta^d}$ for a positive constant d.*

The proof follows analogously to the proof of Theorem 3.3.3.

3.6.3 Constraints on Recipient Anonymity for Unsynchronized Users with Non-compromising Adversaries

Now we shall consider *unsynchronized* user distribution, which is similar to the unsynchronized user distribution for sender anonymity, but with a few changes. Our unified sender has a biased coin corresponding to each recipient with success probability p. In each round, he decides to send a message for a recipient by tossing the biased coin, independent of other recipients as well as other rounds. We denote this user distribution with U_P. We consider a *non-compromising* passive adversary similar to Section 3.6.1.

Theorem 3.6.6. *For user distribution U_P, no protocol $\Pi \in M$ can provide δ-recipient anonymity for any $\delta < 1 - \left(\frac{1}{2} + f_p^{RA}(\ell)\right)$, where $f_p^{RA}(d) = \min\left(\frac{1}{2},\ 1 - (1-p)^{\ell+d}\right)$ and d is an integer ≥ 1.*

Proof. By Claim 6, we know that Π_{ideal} is the optimal protoco for satisfying Invariant 2 and by Claim 4 we know that the invariant is necessary for anonymity. Thus, it suffices to calculate the probability that Π_{ideal} satisfies Invariant 2 as a lower bound of the adversary's advantage against any protocol.

Let, R_0 and R_1 be the recipients chosen by the adversary and let b be the challenge bit. Let s be the round in which the sender sends the challenge message. Since we are considering a non-compromising adversary, the following is true:

$$\Pr[\text{Invariant 2 is true}] = \Pr[T_{RA} \text{ is not empty}].$$

Note that, If a message is sent for the recipient R_{1-b} (enters the protocol) in $[s-\ell, s+\ell-1]$, it has a possibility to populate an element in T_{RA}.

We follow the same calculations as in the proof of Theorem 3.4.1, and derive:

$$\Pr[Y(d)] = 1 - (1 - E(d))(1 - 2f_p(d)),$$

Where $f_p(d)$ is defined as in Theorem 3.4.1, And $Y(d)$ denotes the event that at least one message is sent for a given recipient within an interval of d rounds.

$$\begin{aligned}
&\Pr[T_{RA} \text{ is not empty}] \\
\leq &Y(2\ell) \\
\leq &1 - (1 - E(2\ell))(1 - 2f_p(2\ell)) \\
\leq &1 - \frac{1}{2}(1 - 2f_p(2\ell)) \\
= &1 - \frac{1}{2}\left(1 - 2f_p^{RA}(\ell)\right) = \frac{1}{2} + f_p^{RA}(\ell).
\end{aligned}$$

Hence, $\delta \geq 1 - \Pr[T_{RA} \text{ is not empty}] \geq 1 - \left[\frac{1}{2} + f_p^{RA}(\ell)\right]$. □

Impossibility for Strong Recipient Anonymity. We now investigate under which constraints for ℓ and β Theorem 3.6.1 rules out strong recipient anonymity.

Theorem 3.6.7. *For user distribution U_P and $p > 0$, no protocol can achieve strong anonymity recipient if $2\ell p < 1 - \epsilon(\eta)$, where $\epsilon(\eta) = 1/\eta^d$ for a positive constant d.*

The proof follows analogously to the proof of Theorem 3.4.2.

3.6.4 Recipient Anonymity for Unsynchronized Users with Partially Compromising Adversaries

Now we extend our analysis for recipient anonymity against partially compromising adversaries, with the same user distribution as the previous section.

Theorem 3.6.8. *No protocol $\Pi \in M$ can provide δ-recipient anonymity for the user distribution U_P, when*

$$\delta < \begin{cases} \left[1 - \frac{\binom{\mathsf{c}}{\ell}}{\binom{\mathsf{K}}{\ell}}\right]\left[\frac{1}{2} + f_p^{RA}(\ell)\right] & \mathsf{c} \geq \ell \\ \left(1 - \left[\frac{1}{2} + f_p^{RA}(\ell - \mathsf{c})\right]\right)\left(1 - \left[\frac{1}{2} + f_p^{RA}(\mathsf{c})\right]\left[1 - \frac{1}{\binom{\mathsf{K}}{\mathsf{c}}}\right]\right) & \\ & \mathsf{c} < \ell \end{cases}$$

where $f_p^{RA}(d) = \min\left(\frac{1}{2},\ 1 - (1-p)^{\ell+d}\right)$ *for integer* $d \geq 1$.

Proof. Let R_0, R_1 be the challenge users and let b be the challenge bit. Moreover, let s_0 be the time the challenge message is sent for R_b and let $r = s_0 + t$ be the time it is received by the recipient, where t is the delivery time decided by the oracle O for the challenge message.

As in proofs for Theorems 3.5.1 and 3.6.6, we define $Y(d)$ as the event that at least one message is sent for a given recipient within an interval of d rounds; and we derive:

$$\Pr\left[Y(d)\right] \leq 1 - \tfrac{1}{2}\left(1 - 2f_p(d)\right) = \tfrac{1}{2} + f_p^{RA}(\tfrac{d}{2}).$$

We distinguish two cases, depending on ℓ and c: 1. First, where the number of compromised parties c is at least as large as the maximal latency ℓ. In this case, all parties on the path of the challenge message could be compromised. 2. Second, where all parties on the path of the challenge message can not be compromised. And hence, the analysis focuses on the delivery times of messages for R_{1-b}.

1) Case $\mathsf{c} \geq \ell$. We know, $\ell \geq t$ holds by definition. The invariant is true if and only if R_{1-b} receives at least one message in one of the rounds between $(s_0 + 1)$ and $(s_0 + \ell)$ and for the last of those messages, delivered at time t_{last}, there is at least one non-compromised party on the path between t_0 and t_{last}. Hence,

$$\begin{aligned} &\Pr\left[\text{Invariant 2 is true}\right] \\ &\leq \Pr\left[R_{1-b} \text{ receives at least one message in } [s_0, s_0 + \ell]\right] \\ &\quad \cdot \Pr\left[NOT \text{ all the c parties are compromised}\right] \\ &\leq \Pr\left[Y(2\ell)\right] \cdot \left[1 - \frac{\binom{\mathsf{c}}{\ell}}{\binom{\mathsf{K}}{\ell}}\right] = \left[\frac{1}{2} + f_p^{RA}(\ell)\right]\left[1 - \frac{\binom{\mathsf{c}}{\ell}}{\binom{\mathsf{K}}{\ell}}\right]. \end{aligned}$$

Therefore, $\delta \geq 1 - \left[1 - \frac{\binom{\mathsf{c}}{\ell}}{\binom{\mathsf{K}}{\ell}}\right]\left[\frac{1}{2} + f_p^{RA}(\ell)\right]$.

2) Case $\mathsf{c} \leq \ell$. The probability that all parties on the mutual path of the challenge message and a message for the alternative recipient R_{1-b} are compromised now mainly depends

on the delivery time of the messages for R_{1-b}. We distinguish two sub-cases depending on the oracle's choice for t:

2a) Case $\mathsf{c} \leq t$:

$$
\begin{aligned}
&\Pr\left[\text{Invariant 2 is true}\right] \\
&\leq \Pr\left[R_{1-b} \text{ receives at least one message in } [s_0+\mathsf{c}, s_0+\ell]\right] \\
&\quad + \Pr\left[R_{1-b} \text{ does NOT receive a message in } [s_0+\mathsf{c}, s_0+\ell]\right] \\
&\qquad \cdot \Pr\left[R_{1-b} \text{ receives at least one message in } [s_0, s_0+\mathsf{c}]\right] \\
&\qquad \cdot \Pr\left[NOT \text{ all the c parties are compromised}\right] \\
&\leq \Pr\left[Y(2\ell-\mathsf{c})\right] + \left(1-\Pr\left[Y(2\ell-\mathsf{c})\right]\right) \\
&\qquad \cdot \Pr\left[Y(\ell+\mathsf{c})\right]\left[1-\frac{1}{\binom{\mathsf{K}}{\mathsf{c}}}\right]
\end{aligned}
$$

Therefore, since $\delta \geq 1 - \Pr\left[\text{Invariant 2 is true}\right]$, we have:

$$
\begin{aligned}
\delta &\geq \left(1-\Pr\left[Y(2\ell-\mathsf{c})\right]\right)\left(1-\Pr\left[Y(\ell+\mathsf{c})\right]\left[1-\frac{1}{\binom{\mathsf{K}}{\mathsf{c}}}\right]\right) \\
&\geq \left(1-\left[\frac{1}{2}+f_p(2\ell-\mathsf{c})\right]\right)\left(1-\left[\frac{1}{2}+f_p(\ell+\mathsf{c})\right]\left[1-\frac{1}{\binom{\mathsf{K}}{\mathsf{c}}}\right]\right) \\
&\geq \left(1-\left[\frac{1}{2}+f_p^{RA}(\ell-\mathsf{c})\right]\right)\left(1-\left[\frac{1}{2}+f_p^{RA}(\mathsf{c})\right]\left[1-\frac{1}{\binom{\mathsf{K}}{\mathsf{c}}}\right]\right).
\end{aligned}
$$

2b) Case $t < \mathsf{c}$:

$$
\begin{aligned}
&\Pr\left[\text{Invariant 2 is true}\right] \\
&\leq \Pr\left[R_{1-b} \text{ receives at least one message in } [s_0+\mathsf{c}, s_0+\ell]\right] \\
&\qquad \cdot \Pr\left[NOT \text{ all the t parties are compromised}\right] \\
&\quad + \Pr\left[R_{1-b} \text{ does NOT receive any message in } [s_0, s_0+\ell]\right] \\
&\qquad \cdot \Pr\left[R_{1-b} \text{ receives at least one message in } [s_0, s_0+\mathsf{c}]\right] \\
&\qquad \cdot \Pr\left[NOT \text{ all the t parties are compromised}\right]
\end{aligned}
$$

$$\leq \Pr\left[R_{1-b} \text{ receives at least one message in } [s_0 + \mathsf{c}, s_0 + \ell]\right]$$
$$+ \Pr\left[R_{1-b} \text{ does NOT receive any message in } [s_0, s_0 + \ell]\right]$$
$$\cdot \Pr\left[R_{1-b} \text{ receives at least one message in } [s_0, s_0 + \mathsf{c}]\right]$$
$$\cdot \Pr\left[NOT \text{ all the t parties are compromised}\right]$$

The above event expression is exactly the same as the expression we had in the previous case ($t > \mathsf{c}$). The bound on δ thus follows analogously. □

3.6.5 Impossibility for Strong Anonymity

The bound for δ is in this scenario is exactly similar to the counterpart of sender anonymity results (Section 3.5). Hence, the analysis follows analogously.

3.7 Implications

To put our result into perspective, we discuss whether our trilemma excludes strong anonymity for a few AC protocols from the literature. More precisely, this section exemplarily applies the results from Theorem 3.2.2 and Theorem 3.4.2, i.e., with synchronized and unsynchronized user distributions and a global network-level, non-compromising adversary. As the latency of some AC protocols depends on system parameters, we carefully choose system parameters and make a few simplifying assumptions, which are subsequently described.

This section is solely intended to put our impossibility result into perspectiveby discussing how we estimated the bandwidth β and latency ℓ bounds in the sense of this work. It is not meant and not qualified to be a performance and scalability comparison of the discussed AC protocols, which would have to take many other dimensions into account, e.g., the communication and computation complexity of the servers and the receivers, the computation complexity of the senders and the different kinds of functionalities that are offered by the different AC protocols (e.g., group communication vs. internet-like visitor-webpage communication). Table 3.1 summarizes bounds on the bandwidth β and latency overhead ℓ (in the sense of this work).

Technically, this section considers translations of AC protocols into our protocol model and estimates the latency and bandwidth overhead of these translations. As these transla-

Table 3.1. Latency vs. bandwidth vs. strong anonymity of AC protocols, with the number of protocol-nodes K, number of clients N, and message-threshold T, expected latency ℓ per node, dummy-message rate β.

Protocol	**Latency**	**Bandwidth**	**Strong Anonymity**
Tor [13]	$\theta(1)$	$\theta(1/\mathsf{N})$	impossible
Hornet [14]	$\theta(1)$	$\theta(1/\mathsf{N})$	impossible
Herd [7]	$\theta(1)$	$\theta(\mathsf{N}/\mathsf{N})$	possible
Riposte [6]	$\theta(N)$	$\theta(\mathsf{N}/\mathsf{N})$	possible
Vuvuzula [2]	$\theta(\mathsf{K})$	$\theta(\mathsf{N}/\mathsf{N})$	possible
Riffle [8]	$\theta(\mathsf{K})$	$\theta(\mathsf{N}/\mathsf{N})$	possible
Threshold mix [30]	$\theta(T\mathsf{K})$	$\theta(1/\mathsf{N})$	impossible*
Loopix [1]	$\theta(\sqrt{K}\ell)$	$\theta(\beta)$	possible

* if T in $o(poly(\eta))$

tions do not provide any additional insights, we do not present the full translated protocols but only the abstraction steps. We abstract away the cryptographic instantiation of messages including the bandwidth overhead they introduce over the plaintext. We assume an upper bound on the latency of the protocol and are oblivious to server-side noise (see Claim 2). Moreover, recall that we are only interested in the question whether our trilemma excludes strong anonymity for the ten AC protocols from the literature; hence, we consider the upper bound on the latency and bandwidth overhead for deterministic latency. For randomized latency, such as Loopix [1], we list for simplicity the expected delay as the latency bound.

Low-latency protocols such as Tor [13], Hornet [14], and Herd [7] are low-latency AC protocols, i.e., they immediately forward messages. While Tor and Hornet do not produce asymptotically more than a constant amount of both bandwidth overhead and latency overhead and thus cannot provide strong anonymity, Herd produces dummy traffic linearly proportional to the number of users (bandwidth overhead $\beta \in \theta(\mathsf{N}/\mathsf{N})$), thus the trilemma does *not* exclude strong anonymity for Herd.

Riposte [6] uses secure multiparty computation and a variant of PIR to implement an anonymous bulletin board. Riposte operates in epochs and for each epoch the set of users is public. Hence, Riposte is expected to be run with long epochs to maximize the number of users that participate in an epoch, which leads us to estimating the latency overhead to be

$\ell \in \theta(N)$. To counter traffic analysis attacks, Riposte clients send constant dummy traffic, resulting in a bandwidth overhead of $\beta \in \theta(\mathsf{N}/\mathsf{N})$. Thus, the trilemma does not exclude strong anonymity for Riposte.

Vuvuzela [2] is a mix-net that is tailored towards messengers. Clients communicate by deposing their encrypted messages in one of the mix net nodes. To achieve strong resistance against compromised servers, Vuvuzela takes a path through all servers, resulting in a latency overhead of $\ell \in \theta(\mathsf{K})$ (for K servers). Additionally, Vuvuzela utilizes constant traffic, leading to a bandwidth overhead of $\beta \in \theta(\mathsf{N}/\mathsf{N})$, and has the potential for strong anonymity.

Riffle [8] uses a verifiable mix-net, however not only for messenger communication but also for normal client-server web traffic. Just as Vuvuzela, Riffle also chooses paths that traverse all K servers, leading to $\ell \in \theta(\mathsf{K})$ and if we assume $\mathsf{K} \in \theta(\log(\eta))$, we get $\ell \in \theta(\log(\eta))$. We assume that the clients send dummy traffic up to a constant rate (depending on the user's sending rate p), so we have $\beta \in \theta(\mathsf{N}/\mathsf{N})$ and the potential for strong anonymity.

In a **threshold mix net**, each of the K mix servers waits until it received up to a threshold T many messages before relaying the messages to the next mix, resulting in $\ell \in \theta(T \cdot K)$. Threshold mixes [30] do not provide strong anonymity unless their threshold T is of the order of the number of users N. As such a large threshold are impractical for a large number of users, we judge it impossible to achieve strong anonymity for practical deployments of Threshold mixes.

Loopix [1] is a mix net that combines exponentially distributed delays at each mix-node and dummy messages from each user. Ignoring so-called loop messages (meant to counter active attacks), Loopix naturally enforces our unsynchronised user distribution: the rate at which Loopix clients send messages is the sum of a dummy-message rate (β) and a payload message rate (p), which are system parameters. We assume that the path lengths in Loopix' stratified topology is $\sqrt{K}$ with the number of nodes $K \in \theta(\log(\eta))$. If $\beta + p \geq 1/\sqrt{\eta}$, and if every hop introduces an expected delay of $\ell \geq \frac{\sqrt{\eta}}{\sqrt{K}}$, the expected latency overhead is $\ell = \sqrt{K} \cdot \ell$, in particular $\ell \in \theta(\sqrt{(}\eta))$. We get $(p+\beta)\ell = \frac{1}{\sqrt{\eta}} \cdot \sqrt{\eta} = 1$ and the trilemma does not exclude strong anonymity for Loopix.

3.A Protocol Model Revisited

3.A.1 Validity of the Protocol Model (Contd.)

Lemma 1. *Let* Π *be a protocol* $\in M$ *with* K *parties with parameters* β *and* ℓ*. Then: 1. Messages are delivered within* ℓ *steps. 2. The protocol adds (for the* unsynchronised *case on average) a maximum of* β *noise messages per user per round. 3. Whenever a party in* $\mathcal{S} \cup \mathsf{P}$ *sends a message to another party in* $\mathsf{P} \cup \mathcal{R}$*, the adversary learns that and in which round this happens. 4. For every message that leaves the network (received by R), the adversary additionally learns whether the message is the target message. 5. For every compromised party, the adversary learns the mapping between the input messages and the output messages.*

Proof. Let Π be a protocol $\in M$ with K parties with parameters β and ℓ. We analyze the lemma part by part.

1. Messages are delivered within ℓ steps.
2. The protocol adds (for the *unsynchronized* case on average) a maximum of β noise messages per user per round.
3. The adversary learns that and in which round a party in $\mathcal{S} \cup \mathsf{P}$ sends a message to another party in $\mathsf{P} \cup \mathcal{R}$.
4. For every message that leaves the network (received by R), the adversary additionally learns whether this message is the target message.
5. For every compromised party, the adversary learns the mapping between the input messages and the output messages.

Part (2) of the Lemma holds, since we restrict the user distributions accordingly and since the none of the transitions in the Petri net can create more tokens within the network than it consumes from its input place.

We show the part (1) of the lemma via structural induction over fired transitions of the Petri net. We additionally add to the induction invariant that all tokens that are not in $\mathcal{S}$ have a timestamp for their next transition of $\mathsf{ts} = 1$ and a remaining time of $\mathsf{t_r} > 0$ and there are at least $\mathsf{t_r}$ rounds left in which the token can be delivered.

Induction base: The protocol is initialized and no transitions have happened. Thus, no messages have been sent so far, i.e., there is no message that has not been delivered within

ℓ steps. The only transition that can fire is $T_{\mathcal{S}}$ and for $\ell > 0$, the message introduced into the network in this way does not need to be delivered already ($0 < \mathsf{t_r} = \ell$). Moreover, $T_{\mathcal{S}}$ sets the timestamp of this message token to $\mathsf{ts} = 1$

Induction step: Let tr be any execution trace s.t. the induction invariant is satisfied and let t be an arbitrary possible transition that extends tr to $tr :: t$.

We distinguish two cases for t: In case t is $T_{\mathcal{S}}$, it consumes a token from $P_{\mathcal{S}}$ and puts this token into a place P_i and, by definition we have $\mathsf{t_r} > 0$ and $\mathsf{ts} = 1$. Otherwise, the transition is T_{P_i} for some i and consumes a token from P_i accordingly. By the induction invariant, the token has $\mathsf{t_r} > 0$. If this token has $\mathsf{t_r} - 1 = 0$, the transition delivers the token to R. Otherwise, t decreases $\mathsf{t_r}$ by one (thus fulfilling the condition that there are at least $\mathsf{t_r}$ rounds left in which the token can be delivered) and sets $\mathsf{ts} = 1$. Since every token in any place P_i needs to be consumed in every round, the protocol delivers every message in at most ℓ steps.

Other parts of the lemma: By definition of our Petri net, whenever a transition fires, an element (t, r) is placed into Tokens, containing the public fields of t, such as $t.\mathsf{prev}$ and $t.\mathsf{next}$, as well as the current round number r, which fulfills part (3). Moreover, whenever the transition places the token in R, the adversary can additionally see the field $t.\mathsf{msg}$ and no transition can change the field msg, which allows the adversary to effectively tag and recognize the challenge message and thus fulfills part (4). Finally, if any party P_i is compromised, P_i does not modify the unique (and otherwise freshly sampled) field $t.\mathsf{ID_t}$, which allows the adversary to map incoming and outgoing messages.

Since the transitions discussed here are the only way for messages to be sent to a recipient, the model correctly enforces the conditions from the lemma. □

3.A.2 Polynomial Boundedness of the adversary $\mathcal{A}_{paths}$

Lemma 2. *If number of messages per user per round is polynomially bounded and the anonymity game stops in polynomial time, our adversary $\mathcal{A}_{paths}$ is also polynomially bounded.*

Proof Sketch. Suppose, the number of messages per user per round is bounded by G, and the game stops in Q rounds. Then, total number of messages in the system is bounded

by $(G \times N \times Q)$. Consequently, total number of elements in the set Tokens is bounded by $(G \times N \times Q \times Q)$. In the worst scenario, $\mathcal{A}_{paths}$ will have to scan $(G \times N \times Q \times Q)$ tokens. $\square$

As long as the user distribution ensures that the number of messages per user per round is polynomially bounded, our adversary is polynomially bounded - which is true for both the user distributions that we use. For the synchronized user distribution, number of messages per user per round is $\frac{\mathsf{N}+1}{\mathsf{N}}$; and for the unsynchronized user distribution that is $p \leq 1$.

3.B Unsynchronized Users with Partially Compromising Adversaries: Lower Bound on Adversarial Advantage (contd.)

Proof of Theorem 3.5.1. As in the proofs for Theorems 3.2.1, 3.3.1 and 3.4.1 we calculate the advantage of $\mathcal{A}_{paths}$ against Π_{ideal} to derive a bound against any protocol in our model.

As in the proof for Theorem 3.4.1 we define the random variables $X^{(1)}(x), X^{(2)}(x), \ldots, X^{(\mathsf{N})}(x)$, where $X^{(i)}(x)$ denotes the event of the i^{th} user sending her own message in an interval of x rounds $[a, b]$, with $(b - a) = x$. All $X^{(i)}(x)$ are mutually independent. Note that we here consider intervals x that are not necessarily of size ℓ.

$$X^{(i)}(x) = \begin{cases} 0 \quad with \quad (1-p)^x \\ 1 \quad with \quad (1-(1-p)^x) \end{cases}$$

As before, we make use of the sum $X(x) = \sum_{i=1}^{\mathsf{N}} X^{(i)}(x)$ over all users and calculate the expected value of $X(x)$ as

$$\begin{aligned} \mathbb{E}[X(x)] = \mathbb{E}\left[\sum_{i=1}^{\mathsf{N}} X^{(i)}(x)\right] &= \sum_{i=1}^{\mathsf{N}} \mathbb{E}\left[X^{(i)}(x)\right] \\ &= \mathsf{N}\left(1-(1-p)^x\right) = \mu(x) \end{aligned}$$

Using the Chernoff Bound on the random variable $X(x)$ calculate $\Pr\left[X(x) - \mu(x) \geq \mathsf{N}a\right] \leq \exp(-2a^2\mathsf{N})$, and for $a = \frac{\mu(x)}{\mathsf{N}}$, we define $E(x)$ as :

$$\begin{aligned} E(x) = \Pr\left[X(x) \geq 2\mu(x)\right] &\leq \exp\left(-2\mu(x)^2/\mathsf{N}^2 \cdot \mathsf{N}\right) \\ &\leq \exp\left(-2(1-(1-p)^x)^2\mathsf{N}\right). \end{aligned}$$

Note that, similar to $X^{(i)}(x)$ and $X(x)$, $\mu(x)$ is also defined as in the proof for Theorem 3.4.1, but for a slice of variable length x. We denote the event that sender u_{1-b} sends at least one message in an interval of size x by $Y(x)$ and since all users are acting independently from each other we get for $j \in \{0, \ldots, \mathsf{N}\}$, $\Pr[Y(x)|X(x) = j] = 1 - \Pr[\neg Y|X(x) = j] = \frac{j}{\mathsf{N}}$. Moreover, for any value of x with $2\mu(x) \leq \mathsf{N}$,

$$\begin{aligned}
\Pr[Y(x)] &= \Pr[X(x) \geq 2\mu(x)] \cdot \Pr[Y(x)|X(x) \geq 2\mu(x)] \\
&\quad + \Pr[X(x) < 2\mu(x)] \cdot \Pr[Y(x)|X(x) < 2\mu(x)] \\
&\leq \Pr[X(x) \geq 2\mu(x)] \cdot \Pr[Y(x)|X(x) = \mathsf{N}] \\
&\quad + \Pr[X(x) < 2\mu(x)] \cdot \Pr[Y(x)|X(x) = 2\mu(x)] \\
&= E(x)\Pr[Y|X(x) = \mathsf{N}] \\
&\quad + (1 - E(x)) \Pr[Y|X(x) = 2\mu(x)] \\
&= E(x) \left({}^{\mathsf{N}}/_{\mathsf{N}}\right) + (1 - E(x)) \left({}^{2\mu(x)}/_{\mathsf{N}}\right) \\
&= 1 - (1 - E(x)) \left(1 - 2\left(1 - (1-p)^x\right)\right).
\end{aligned}$$

If $2\mu(x) > \mathsf{N}$, we get with $f(x) = \min\left(\frac{1}{2},\ 1 - (1-p)^x\right)$:

$$\begin{aligned}
\Pr[Y(x)] &\leq E(x) + (1 - E(x)) \cdot 1 \leq 1 \\
&\leq 1 - (1 - E(x))\,(1 - 2f(x))\,.
\end{aligned}$$

Now, we calculate the probability of Invariant 1 being true, under our protocol Π_{ideal} and as in the proof for Theorem 3.3.1. We distinguish two cases depending on c and ℓ:

Case 1): $\mathsf{c} > \ell$

$$\begin{aligned}
&\Pr[\text{Invariant 1 is true}] \\
&\leq \Pr[\neg\text{Cmpr}(\ell)] \cdot \Pr[u_{1-b}.\text{sent}(r-\ell, r-1)] \\
&= \Pr[\neg\text{Cmpr}(\ell)] \cdot \Pr[Y(\ell)] \\
&\leq \left[1 - \tbinom{\mathsf{c}}{\ell} / \tbinom{\mathsf{K}}{\ell}\right] \left[1 - \left(1 - E(\ell)\right)\left(1 - 2f_p(\ell)\right)\right].
\end{aligned}$$

By applying Markov's inequality on the random variable $X(x)$, we get $E(x) = \Pr[X(x) \geq 2\mu(x)] \leq \frac{1}{2}$. Thus, we derive for δ: $\delta \geq 1 - \left[1 - \binom{\mathsf{c}}{\ell}/\binom{\mathsf{K}}{\ell}\right]\left[\frac{1}{2} + f_p(\ell)\right]$.

Case 2): $\mathsf{c} < \ell$**.** As for the proof of Theorem 3.3.1 we split this case into two sub-cases, depending on t and c.

Case 2a): $\mathsf{c} < t$

$$
\begin{aligned}
&\Pr[\text{Invariant 1 is true}] \\
\leq\ &\Pr[u_{1-b}.\text{sent}(r-\ell, r-\mathsf{c})] + \Pr[\neg u_{1-b}.\text{sent}(r-\ell, r-\mathsf{c})] \\
&\qquad \cdot \Pr[u_{1-b}.\text{sent}(r-\mathsf{c}, r)] \cdot \Pr[\neg\text{Cmpr}(\mathsf{c})] \\
=\ &\Pr[Y(\ell-\mathsf{c})] + [1 - \Pr[Y(\ell-\mathsf{c})]] \Pr[Y(\mathsf{c})] \Pr[\neg\text{Cmpr}(c)] \\
\leq\ &[1 - (1 - E(\ell-\mathsf{c}))(1 - 2f_p(\ell-\mathsf{c}))] \\
&\quad + [(1 - E(\ell-\mathsf{c}))(1 - 2f_p(\ell-\mathsf{c}))] \\
&\qquad \cdot [1 - (1 - E(\mathsf{c}))(1 - 2f_p(\mathsf{c}))]\left[1 - {}^1/\binom{\mathsf{K}}{\mathsf{c}}\right].
\end{aligned}
$$

Thus, for the adversarial advantage δ we derive,

$$
\begin{aligned}
\delta \geq\ &1 - \Pr[\text{Invariant 1 is true}] \\
\geq\ &1 - [1 - (1 - E(\ell-\mathsf{c}))(1 - 2f_p(\ell-\mathsf{c}))] \\
&\quad - [(1 - E(\ell-\mathsf{c}))(1 - 2f_p(\ell-\mathsf{c}))] \\
&\qquad \cdot [1 - (1 - E(\mathsf{c}))(1 - 2f_p(\mathsf{c}))]\left[1 - \binom{\mathsf{c}}{\mathsf{c}}/\binom{\mathsf{K}}{\mathsf{c}}\right] \\
=\ &[(1 - E(\ell-\mathsf{c}))(1 - 2f_p(\ell-\mathsf{c}))] \\
&\quad \cdot \left(1 - [1 - (1 - E(\mathsf{c}))(1 - 2f_p(\mathsf{c}))]\left[1 - 1/\binom{\mathsf{K}}{\mathsf{c}}\right]\right) \\
\geq\ &\left(1 - \left[\frac{1}{2} + f_p(\ell-\mathsf{c})\right]\right)\left(1 - \left[\frac{1}{2} + f_p(\mathsf{c})\right]\left[1 - 1/\binom{\mathsf{K}}{\mathsf{c}}\right]\right).
\end{aligned}
$$

We again use Markov's inequality to replace $E(x)$ by $1/2$.

Case 2b): $t \leq \mathsf{c}$

$$
\begin{aligned}
&\Pr\left[\text{Invariant 1 is true}\right] \\
&\leq \Pr\left[u_{1-b}.\text{sent}(r-\ell, r-\mathsf{c})\right] \cdot \Pr\left[\neg\text{Cmpr}(t)\right] \\
&\quad + \Pr\left[\neg u_{1-b}.\text{sent}(r-\ell, r-\mathsf{c})\right] \\
&\qquad \cdot \Pr\left[u_{1-b}.\text{sent}(r-\mathsf{c}, r)\right] \cdot \Pr\left[\neg\text{Cmpr}(\mathsf{c})\right] \\
&\leq \Pr\left[u_{1-b}.\text{sent}(r-\ell, r-\mathsf{c})\right] + \Pr\left[\neg u_{1-b}.\text{sent}(r-\ell, r-\mathsf{c})\right] \\
&\qquad \cdot \Pr\left[u_{1-b}.\text{sent}(r-\mathsf{c}, r)\right] \Pr\left[\neg\text{Cmpr}(\mathsf{c})\right]
\end{aligned}
$$

The above event expression is exactly same as the expression we had in the previous case ($t > \mathsf{c}$). Thus, the rest of the calculations and bounds are exactly same as the previous case. □

4. TRILEMMA FOR AC PROTOCOLS WITH USER COORDINATION

In the last chapter we presented an impossibility result for sender and recipient anonymity of ACNs that allow messages to be sent, to mix, and to confuse a potential adversary with dummy messages. However, in this chapter we are going to identify a class of protocol that can escape those impossibility results by using a technique, what we call out-of-band *user coordination*. We start by showing, with an intuitive counter-example, why the anonymity trilemma proven in the last chapter does not sufficiently capture this space of protocols.

Imagine a protocol in which users communicate out-of-band to initialize secret-sharing for their messages, e.g., they use a technique leveraged by DC-nets [11] with pre-setup key agreement, where each user only needs to publish their local messages. Whenever a recipient receives a set of messages that belong together, the recipient has to combine all of them to extract the real message. There is a certain chance that when Alice sends her message, Bob is one of the users who provides a share. In this instance, no matter the level of compromisation or the latency overhead of the protocol, the adversary won't be able to know who out of Alice or Bob actually initiated the message.

This property was not captured in the previous chapter; consequently, we may ask whether such techniques give us hope for cheap strong anonymity. We now set out to formally show that this hope is unfounded (or, at the very least, requires stronger techniques than currently available). We use the term shares to refer to messages created to confuse the adversary in such a way, and use the term user coordination to refer to the process. By these terms we do not refer to specific techniques, but rather capture all sorts of techniques that lead to this effect.

About Recipient Anonymity. We note that for recipient anonymity shares do not help, since recipient anonymity is defined as a property of tracking individual packets. While shares can obfuscate the real sender among several users, they do not prevent the tracking of any individual packet to a recipient. Therefore, recipient anonymity bounds from Chapter 3 do not change with user coordination, and in this chapter we focus on the scope of improvements of sender anonymity.

4.1 Assumptions on user coordination

We impose the following assumptions on user coordination:

1. If $h + 1$ shares are used to reconstruct a message, at least one of them is sent by the original sender.

2. no share can take part in reconstructing two separate messages.

3. A compromised protocol party is always able to map its outgoing packets to its incoming packets.

Our assumptions are consistent with most ACNs from the literature. In Section 4.1.1, we discuss how breaking these assumptions presents cryptographic challenges.

We do not restrict our protocols to any specific technique of UC; it can be any method (e.g., secret sharing, multi-party computation etc.) that achieve the user coordination property. However, for our impossibility analysis, we assume that user coordination can be achieved via a pre-processing step or can be done efficiently, and hence, we ignore the cost of user coordination.

Additionally, to be consistent with how we count the latency overhead for a real message, we add the restriction that every packet (real or noise, created by a user or an internal protocol party) is allowed to remain in the system for no more than ℓ rounds. Finally, if a message is scheduled to be sent in round t_0 by the user distribution, all shares of that message (as well as the real packer) must reach the recipient before round $t_0 + \ell$.

4.1.1 Discussion about user coordination assumptions

In Section 3.1, we make three assumptions regarding the protocol model. Most ACNs from the literature are consistent with these assumptions.

The first assumption is that among the shares employed to reconstruct a message at least one must be sent by the message sender. This follows from our assumption that the messages are unavailable while User Coordination gets established; if senders were allowed to know and transmit their messages during setup, the whole protocol could take place during the setup phase.

The second assumption is that no share can take part in reconstructing two separate messages. Although concepts such as Ramp secret-sharing [50] from the cryptographic literature indeed offer the possibility to extract multiple shared messages from a given set of shares, it requires messages to be known in advance. In general, reusing the same share will introduce confidentiality issues similar to a two-times pad. Dicemix [15] uses such a technique where shares of different messages are mixed and is thus outside our model. Nevertheless, Dicemix utilizes n^2 shares for reconstructing n messages; so, it does not break our impossibility bounds. Recent mailbox-based schemes like Riposte [6] are within our protocols model.

The third assumption is indeed interesting. It expects that a compromised party will always be able to map its outgoing packets with its incoming packets. Although this is trivially correct when there is one incoming packet, the assumption focuses on the question when there are two or more incoming packets. It suggests that the party cannot permute these multiple incoming packets such that it itself cannot determine the employed permutation. Performing non-interactive MPC using fully homomorphic encryption (FHE) [51] may enable a node to permute message locally (i.e., without introducing bandwidth and latency overheads) without determining the permutation. This is highly inefficient for current FHE mechanisms as the evaluation circuit depth will be at least logarithmic in the number of users. Nevertheless, it presents an interesting avenue for future ACN design.

4.2 Updated Protocol model for AC protocols

In this section we discuss the new protocol model that can capture protocols with user coordination, along with all the protocols that we considered in Chapter 3.

The main purpose of an AC protocol is to let an AC-user (from the set of users $\mathcal{S}$) send information to a recipient (from the set of recipients $\mathcal{R}$). Typically, an AC protocol utilizes a set of nodes (anonymizing parties P) to improve performance and distribute trust. Similar to Chapter 3, we consider a global eavesdropping (i.e., passive) adversary $\mathcal{A}$ that can observe the link between any two parties $\mathcal{S} \cup \mathsf{P}$ (including anonymizing parties and users) and has additionally compromised a set of c anonymizing parties $\mathsf{P_c} \subseteq \mathsf{P}$.

We assume that the AC protocol uses cryptographic means (e.g., encryption or secret sharing) to hide the actual message that a *packet* between two parties $P_1, P_2 \in \mathsf{P} \cup \mathcal{S}$ contains. We abstract the leakage of each such packet as the current round number, the direct sender P_1, the direct recipient P_2, and a random identifier for the packet. This leakage indicates that a packet was sent but doesn't leak any content. Consequently, the adversary only sees the challenge message in plain text when it reaches the recipient.

We stress that we do not require the sets $\mathcal{S}$, $\mathcal{R}$, and P to be mutually disjoint. In some protocols from the literature, these sets actually intersect [11, 12, 15]. As we concentrate on sender anonymity, for simplicity we require the set $\mathcal{R}$ of recipients, to be disjoint from $\mathcal{S} \cup \mathsf{P}$.

Next, using a Petri net model, we formally define a generic AC protocol that captures a large class of AC protocols. This section presents an extension of the protocol model from Chapter 3 with User coordination. Hence, large parts of the protocol model coincide with the protocol model from Chapter 3.

4.2.1 Protocol model

This section defines a generic timed colored Petri net [40, 41, 46] M that can be instantiated with a large class of (abstractions of) AC protocols from the literature. We use K as set of parties, $\mathcal{S}$ as the set of users, $P_1, \ldots, P_{\mathsf{K}}$ as the anonymizing (protocol) parties, \$1 as the randomness, R as the recipient of messages, m as a message or packet (containing a real user message, a noise, or being a share) sent by a client or a protocol parties, $T_{\mathcal{S}}$ as transitions for inserting messages into the Petri net (i.e., into the AC protocol), and $T_{P_1}, \ldots, T_{P_{\mathsf{K}}}$ as transitions for sending messages from one party to another. We stress that for every AC protocol, we use the same Petri net M, i.e., the same places, tokens, and transitions. The guards within the transitions can, however, differ; hence, instantiating this generic Petri net M for (the abstraction of) a concrete AC protocol amounts to specifying the guards within the transitions, e.g., by specifying to which party messages are sent next or how much a message should be delayed. As this specification of the generic Petri net M shows, all protocols that can be instantiated by M, in particular these guards, are oblivious to the challenge message or the challenge users.

Next, we introduce the abstraction of packets in our Petri net model. Formally, packets are colored tokens with eight components. Four public components that an adversary can observe are a unique identifier $\mathsf{ID_t}$, the sender prev and the receiver next of a packet, and the time ts at which the packet is activated. The four private components that an adversary cannot observe are the message content msg, some internal protocol meta-data meta, the message's time-to-live $\mathsf{t_r}$, and the share-group tag to which this message belongs (see below).[1]

Definition 4.2.1 (Colored token)**.**

A colored token *is represented by the tuple* $m = \langle \mathsf{msg},\ \mathsf{tag}, \mathsf{meta}, \mathsf{t_r},\ \mathsf{ID_t},\ \mathsf{prev},\ \mathsf{next},\ \mathsf{ts} \rangle$*, where,*

- msg *is the content of the message,*
- meta *is the internal protocol meta-data for this message,*
- $\mathsf{t_r}$ *is the time the message can remain in the network,*
- $\mathsf{ID_t}$ *is a new unique ID generated by each transition for each token by honest parties; dishonest parties instead keep* $\mathsf{ID_t}$ *untouched to allow the adversary to link incoming and outgoing messages,*
- prev *is the party/user that sent the token and* next *is the user/party that receives the token.*
- *Finally,* ts *is the time remaining for the token to be eligible for a firing event (a feature of timed Petri nets). Here,* ts *either describes when new messages are introduced into the Petri net or is set to the next round, such that messages can be processed in every round as soon as they enter the network.*
- *For allowing user coordination, we introduce an additional field* tag *that allows a token to be* tagged *and several such tokens to contribute to sending one single message. When user coordination is used by the protocol,* msg *field of all the tokens contributing for a single message are populated with* $\perp$*, and the* tag *field of all those tokens are populated with a same tag. We discuss below, how the recipient can retrieve the original message content once he receives a sufficient number of such tokens.*

[1] As sender anonymity solely considers one recipient, for simplicity we do not list the final recipient of the message in the private part.

$\mathsf{ID_t}, \mathsf{prev}, \mathsf{next}, \mathsf{ts}$ are public fields – which means they are always visible to the adversary. However, the fields meta and $\mathsf{t_r}$ are never visible to the adversary. The fields msg and tag can not be observed by the adversary until a packet reaches the recipient.

In case user coordination is used, the field msg does not help to retrieve the message content (because $\mathsf{msg} = \perp$). In this case we use a more complex reconstruction: the recipient has access to a dictionary D (outside the petri-net); when a message reaches the recipient, the recipient queries the dictionary D to retrieve the content of the message. The dictionary has four fields $\langle \mathsf{tag}, \mathsf{msg}, \texttt{count}, \texttt{countNeeded} \rangle$. The field msg stores the actual content of the message. The fields $\mathsf{tag}, \mathsf{msg}, \texttt{countNeeded}$ are already populated (during initialization of the system), whereas the value of $\texttt{count}$ is set to 0 initially. Every time, the recipient queries the dictionary with $\mathsf{D}[\mathsf{tag}]$, the dictionary increments the value of $\texttt{count}$ by 1; and only when $\texttt{count}$ reaches the value of $\texttt{countNeeded}$ it returns msg. We want to specify here that each token in our petri-net model can contain only one tag.

We define a set Tokens that that contains each pair (t, r), where t is a copy of a colored token and r the round number in which the token was observed. Formally, we introduce a set Tokens, that is initially empty and in which we collect the pair (t, r), where t is a copy of a token and r the round number in which the token was observed.

Places. Any AC protocol with K parties $\mathsf{P} = \{P_1, \ldots, P_{\mathsf{K}}\}$ consists of the following places:

- $\mathcal{S}$: A token in $\mathcal{S}$ denotes a user message (real or noise) which is scheduled to enter the network after ts rounds.
- \$1: This place is responsible for providing randomness. Whenever a transition picks a token from this place, the transition basically picks a random value.
- P_i with $P_i \in \mathsf{P}$: A token in P_i denotes a message which is currently held by the party $P_i \in \mathsf{P}$.
- R: A token in R denotes a message which has already been delivered to a recipient.

Transitions. At the beginning of the execution, the challenger specifying the set $\mathcal{S}$ on behalf of the AC protocol. The other places are initialized as empty. Transferring a message from one party to another party is formalized by executing a transition that modifies the *configuration* of the Petri net by consuming a token from one place to producing a token in

Transitions in the Petri net model M

$\mathbf{T_X}$ on tokens $q = \langle \mathsf{msg}, \mathsf{tag}, \_, \mathsf{t_r}, \mathsf{ID_t}, \_, \mathsf{prev}, \mathsf{ts} \rangle$ **from** $X \in \mathcal{S} \cup \mathsf{P}$, \$ **from** \$1**:**
  $(P, \mathsf{meta}) = f_\Pi(q, \$)$; r = current round
  if $\mathsf{t_r} = 0$ **then** $P = R$
  if $X \in \mathsf{P}$ and X is compromised **then** $\mathsf{ID_t} = \mathsf{ID_t}$
  else $\mathsf{ID_t}$ = a fresh randomly generated ID
  $t = \langle \mathsf{tag}, \mathsf{meta}, \mathsf{t_r} - 1, \mathsf{ID_t}, P_i, P, 1 \rangle$
  if $P \neq R$ **then** obs $= \langle \_, \_, \_, \_, \mathsf{ID_t}, \mathsf{prev}, P, 1 \rangle$
  else obs $= \langle \mathsf{msg}, \mathsf{tag}, \_, \_, \mathsf{ID_t}, \mathsf{prev}, P, 1 \rangle$
  $\mathsf{Tokens} = \mathsf{Tokens} \cup \{(\text{obs}, r)\}$
Output: token t at P

f_Π: The code for this function is provided by protocol Π. It decides to which party the message is sent next, as well as the content of the meta field in the token.

Reconstruct(tag)**:**

  if tag $= \perp$ or D[tag] does not exist **then return** $\perp$
  D[tag].count = D[tag].count + 1
  if D[tag].count = count.countNeeded **then return** D[tag] **else return** _

Figure 4.1. Transitions in the Petri net model M

another place. The Petri net M includes the following transitions, for which the Figure 4.1 presents the pseudocode.

- $T_{\mathcal{S}}$: takes a token $\langle \mathsf{msg}, \mathsf{tag}_{\_,\_,\_,\_}, u, \mathsf{ts} \rangle$ from $\mathcal{S}$ and a token from \$1 to write $t = \langle \mathsf{msg}, \mathsf{tag}, \mathsf{meta}, \ell, \mathsf{ID_t}, u, P_i, \mathsf{ts} = 1 \rangle$ to P_i; the values of i and meta are decided by the AC protocol.
- T_{P_i}: takes a token $\langle \mathsf{msg}, \mathsf{tag}, \mathsf{meta}, \mathsf{t_r}, \mathsf{ID_t}, \_, P_i, \mathsf{ts} \rangle$ from P_i and a token from \$1 to write $t = \langle \mathsf{msg}, \mathsf{tag}, \mathsf{meta}, \mathsf{t_r} - 1, \mathsf{ID_t}, P_i, P, 1 \rangle$ to P. If P_i is an honest party $\mathsf{ID_t}$ is freshly generated, but if P_i is a compromised party $\mathsf{ID_t} = \mathsf{ID_t}$. The place $P \in \{P_1, \ldots, P_\mathsf{K}\} \cup \{R\}$ and meta are decided by the AC protocol, except when $\mathsf{t_r} = 0$, P always is R.

The execution of each transition is followed by adding a pair (t, r) to the set Tokens, with t being a copy of the produced token t without the fields meta and $\mathsf{t_r}$ and r being the current round number. Moreover, if the place where t was produced is not in $\mathcal{R}$ also the field msg is not contained in t.

4.2.2 Game Setting

The game setting is similar to that of Chapter 3 with minor modifications. We consider the following game between a PPT adversary $\mathcal{A}$ and an honest challenger $\mathtt{Ch}(\Pi, \alpha_{SA}, b)$:

- $\mathcal{A}$ compromises up to c parties from P.
- $\mathcal{A}$ chooses two distinct users u_0 and u_1 as challenge users. $\mathcal{A}$ sends a challenge message $(\mathtt{Chall}, u_0, u_1, R, R, m^*)$ for those chosen users.
- $\mathtt{Ch}$ then runs protocol Π on (u_b, R, m^*). Π is executed in two parts, Π_{wrapper} and Π_{core}, as described below. (We refer to Figure 4.2 for the pseudocode of Π.)
- First, Π_{wrapper} generates tokens following the user distribution and embeds the challenge message (u_b, R, m^*) in the tokens. Π_{wrapper} adds all the tokens to the place $\mathcal{S}$. Π_{wrapper} does not pass any information about the challenge user or the challenge message to Π_{core}.
- We also allow Π_{core} to add noise tokens to $\mathcal{S}$, limited by the protocol's restrictions on bandwidth overhead. After that, Π_{core} runs the petri net with protocol specific implementation of the transitions.
- For each element in Tokens, $\mathcal{A}$ can see the round number as well as the public parts of the token $(\mathsf{ID_t}, \mathsf{prev}, \mathsf{next}, \mathsf{ts})$, but the private parts $(\mathsf{msg}, \mathsf{meta}, \mathsf{t_r})$ are not communicated to the adversary (c.f. Figure 4.1). However, when the field next is the recipient, the msg field is not obfuscated.
- The goal of the adversary is to deanonymize the sender of the challenge message, i.e., to learn whether the challenge message was sent by u_0 or by u_1. The interaction between $\mathtt{Ch}$ and $\mathcal{A}$ ends as soon as $\mathcal{A}$ makes a guess.

4.3 Towards a new trilemma

With the protocol model in place and anonymity defined, we can now investigate the fundamental limitations of protocols. To this end, we define an abstract protocol within our

Run protocol Π on $r = (u^*, R^*, m^*)$

Π on $r = (u^*, R^*, m^*)$ and user distribution U:

$I_U \leftarrow \Pi_{\text{wrapper}}(r, U)$,
where I_U is a set and each element in I_U is a tuple (u, R, m, ts).
Run $\Pi_{\text{core}}(I_U, U, \beta)$.

Π_{wrapper} on $r = (u^*, R^*, m^*)$ and user distribution U:

generate a set I_U following U,
where I_U is a set and each element in I_U is a tuple (u, R, m, ts).
$e = (u, R, m, \mathsf{ts}) \xleftarrow{\$} I_U$ such that $u = u^* \wedge R = R^*$.
if e exists **then**
  $I_U \leftarrow \{(u^*, R^*, m^*, \mathsf{ts})\} \cup I_U \setminus \{e\}$
for each element $e = (u, R, m, \mathsf{ts})$ in I_U **do**
  Add a token $t = \langle m, \_, \_, \_, u, \_, \mathsf{ts} \rangle$ in the place $\mathcal{S}$.
Output: I_U

Π_{core} on I_U, U, β:

Add tokens in the place $\mathcal{S}$ within the limit of β noise messages per user per round.
Run the petri net with the protocol specific implementation of f_Π.

Figure 4.2. Description of protocol Π

model that leverages user coordination combined with mixing techniques. We then show that this protocol can achieve a better degree of anonymity than the classical impossibility results of that we proved in Chapter 3.

The intuitive reason for this effect is that such a protocol introduces an additional form of uncertainty for the adversary that was not captured by the classical impossibility results. Imagine an adversary that compromises every node in the path that a particular packet traverses and then observes that the packet is being used to reconstruct a message. This adversary might not always learn who actually sent the reconstructed message: all the packets with shares that belong together have to be combined to learn the respective message sent in that particular round; thus all potential senders of these packets could be the message's sender.

We then show an anonymity trilemma that captures even protocols with user coordination: every protocol in our model can be defeated by a straight-forward path adversary unless the protocol utilizes sufficient bandwidth and latency overhead that depends on the degree of compromisation in the network.

4.3.1 AC leveraging user coordination:

We now describe a protocol that falls within our protocol model (Section 3.1) and that leverages user coordination to provide more anonymity than postulated in the impossibility results of Chapter 3 for some values of ℓ, β, and c. While this doesn't show that their result is wrong (they didn't consider user coordination), it emphasizes the importance of covering such protocols in an impossibility result.

The main idea that allows this to work is that we use our bandwidth overhead for shares. Each such share is associated with one real message (with content) within the system and the recipient needs to collect all the shares of a message to decipher it. When all the shares of a message reach a recipient, the adversary can thus only learn that the message has reached and which packets were involved in reconstructing it, but not point to one specific packet it was in.

We assume that the adversary can not break the sharing of message origin provided by user coordination and hence can not decipher an individual message before it reaches the recipient. Additionally, we assume that our user coordination happens out-of-band and is efficient. (For instance, in DC-net [11] with pre-setup key agreement, the protocol parties only need to publish their local messages.) The protocol works in the following way:

1. Users send messages based on a given user message distribution (c.f., Chapter 2).
2. All users participate in the out-of-band user coordination. Instead of sending a dummy noise message, users send shares for other users' messages.
3. All users together run an out-of-band consensus protocol to decide when their messages (real message or share) will be delivered, such that in a given round the recipient receives shares of the same message and all the shares of that message (comparable to t-out-of-t secret sharing).

4. In a given round, the recipient combines all the shares that he receives to extract the real message.
5. The protocol utilizes a series of up to K relays; as long as messages (real or share) are in the system, they are sent from one relay to the next. Note the attacker can compromise up to c of these relays. To prevent the attacker from compromising a consecutive series of relays, we permute the order in which relays are being used. Once the protocol starts, the sequence of the relays is sent to all users.

Analysis of adversarial advantage for the above protocol. We know from Chapter 3 that for the *synchronized user distribution*, the adversarial advantage δ should be lower bounded by

$$\delta \geq 1 - \left[1 - \binom{\mathsf{c}}{\ell} / \binom{\mathsf{K}}{\ell}\right] \times min\left(1, \tfrac{\ell + B\ell}{\mathsf{N}-1}\right).$$

Recall that according to the sender anonymity notion (c.f., Section 2.1), the adversary has to distinguish between two potential senders of a message, u_0 and u_1. If, say, u_0 sends the challenge message and the adversary has compromised every entity on the path this message takes from u_0 to the recipient, then the classical trilemma insists that the adversary wins, which is correct for protocols without user coordination. With user coordination, however, it is possible that u_1 sends a share of the challenge message. This occurs with probability $\frac{B}{\mathsf{N}-1}$. If this happens, there is no way in which the adversary can know whether u_0 or u_1 has sent the challenge message (even if the whole path was compromised) and hence $\delta \leq 1 - \frac{B}{\mathsf{N}-1}$ must hold.[2] We directly see a conflict between the anonymity achieved by our protocol and the impossibility result. For an illustrative example consider the case where $\ell = 1, \mathsf{K} = 2, \mathsf{c} = 1$. We compare the upper bound on δ derived directly from user coordination with the lower bound from Chapter 3 and yield:

$$\begin{aligned}
& 1 - \frac{B}{\mathsf{N}-1} < 1 - \left[1 - \binom{\mathsf{c}}{\ell} / \binom{\mathsf{K}}{\ell}\right] \times min\left(1, \frac{\ell + B\ell}{\mathsf{N}-1}\right) \\
\Longleftarrow\ & \frac{B}{\mathsf{N}-1} > \frac{1}{2} \times \frac{\ell + B\ell}{\mathsf{N}-1} && \text{assuming } \frac{\ell + B\ell}{\mathsf{N}-1} < 1 \\
\Longleftrightarrow\ & 2B > 1 + B \iff B > 1.
\end{aligned}$$

[2] ↑Note that the setup for the user coordination can happen out of band and thus doesn't add any bandwidth overhead here.

Thus, with just one noise message per real message ($B = 1$), our protocol violates the classical impossibility bounds. More generally, if a set of users sends shares for a given message, the adversary can not distinguish the actual sender of the message from other users in the set, unless the user coordination is broken. Note that this effect is similar to the amount of uncertainty introduced by messages meeting (and mixing) in an honest relay.

4.3.2 The path possibility adversary

We use the same adversary $\mathcal{A}_{paths}$ as in Chapter 3 with minor modifications to incorporate user coordination. As we consider sender anonymity, the adversary can start its analysis of all observations from the challenge message that it observes at the recipient. The adversary $\mathcal{A}_{paths}$ constructs all possible paths from which the challenge message could have originated. Recall that in the sender anonymity game the adversary knows two candidate senders u_0 and u_1. So, the adversary checks whether there is a possible path from the challenge message to u_0 and to u_1. If there is no path to one of then, say u_b, the adversary chooses the other challenge sender u_{1-b}. If there is a path to both of them, $\mathcal{A}_{paths}$ makes a random choice.

More precisely, let $(t, r) \in \mathsf{Tokens}$ be an adversary observation, with t being the colored token that was observed in round r. Let r be the round at which the challenge message arrives. Fix $j \in \{0, 1\}$, and let a *possible path for* u_j be a path from a challenge user u_j to the recipient R such that the path is at most ℓ elements long. Observe that if the challenge bit is b the there is at least one possible path for u_b; there has to be a path from u_b to the recipient R.

$$
\begin{aligned}
S_j = \{ & p = (t_1.\mathsf{prev}, \ldots, t_k.\mathsf{prev}, t_k.\mathsf{next}) : \\
& ((t_1, r_1), \ldots, (t_k, r_k)) \in \mathsf{Tokens} \text{ s.t. } k \leq \ell \\
& \wedge\, t_1.\mathsf{prev} = u_j \wedge t_k.\mathsf{next} = R \\
& \wedge\, (t_k.\mathsf{msg} = \texttt{Chall} \vee \mathsf{D}[t_k.\mathsf{tag}] = \texttt{Chall}) \\
& \wedge\, \forall_{i \in \{1,\ldots,k-1\}} (t_i.\mathsf{next} = t_{i+1}.\mathsf{prev} \wedge r_{i+1} = r_i + 1 \\
& \wedge\, (\ \exists t_{i+1} : (t_{i+1}, r_{i+1}) \in \mathsf{Tokens} \wedge t_{i+1}.\mathsf{prev} = t_i.\mathsf{next} \\
& \wedge\, t_{i+1}.\mathsf{ID_t} = t_i.\mathsf{ID_t}) \Rightarrow t_{i+1} = t_{i+1}) \}
\end{aligned}
$$

Definition 4.3.1 (Adversary $\mathcal{A}_{paths}$)**.** *Given a set of users $\mathcal{S}$, a set of protocol parties P of size K, and a number of possibly compromised nodes c, the adversary $\mathcal{A}_{paths}$ proceeds as follows:*

1. *$\mathcal{A}_{paths}$ selects and compromises c different parties from P uniformly at random.*
2. *$\mathcal{A}_{paths}$ chooses two challenge users $u_0, u_1 \in \mathcal{S}$ uniformly at random.*
3. *$\mathcal{A}_{paths}$ makes observations and, based upon those, constructs the sets S_0 and S_1. For any $i \in \{0, 1\}$, if $S_i = \emptyset$, then $\mathcal{A}_{paths}$ returns $1 - i$. Otherwise, it returns 0 or 1 uniformly at random.*

We stress that $\mathcal{A}_{paths}$ does (per run) not take any probabilities into account. Even if in a particular run it is overwhelmingly more likely that u_b sent the message but there is a negligible chance that u_{1-b} sent the message, $\mathcal{A}_{paths}$ does not decide for either of them and randomly picks one. Moreover, if $\mathsf{c} = 0$, $\mathcal{A}_{paths}$ only constitutes a *non-compromising* global network-level adversary, which compromises no protocol parties yet listens on all links between nodes. If $\mathsf{c} > 0$, $\mathcal{A}_{paths}$ is a *partially compromising* global network-level adversary.

4.3.3 New Necessary invariant for anonymity

Invariant 1 defined in Chapter 3 to prove the first trilemma is not suitable anymore. Critically, this invariant *is not necessary* for protocols with user coordination (see our example above in Section 4.3.1). We now derive a new invariant that remains necessary for anonymity in the presence of protocols with user coordination.

Invariant 3 (New Invariant)**.** *Let u_0 and u_1 be the challenge users; let b be the challenge bit. Assume that the challenge message reaches the recipient at time r. Assume furthermore that u_{1-b} sends her messages (including noise messages) at $V = \{t_1, t_2, t_3, \dots\}$. If $T = \{t : t \in V \wedge (r - \ell) \leq t < r\}$,*

(i) the set T is not empty, AND

(ii) (a) at least one share of the challenge message is dispatched by u_{1-b} within the rounds $\{(r - \ell), \dots, (r - 1)\}$, OR

(b) at least one share of the challenge message passes through an honest node at time t such that $t \in \{min(T), (r-1)\}$, AND at least one of the messages (real message or noise) from u_{1-b}, sent at $t \in \{(r-\ell), \ldots, (r-1)\}$, passes through an honest node at time t such that $t < r$.

Claim 7 (Invariant 3 is necessary for anonymity)**.** *Let Π be any protocol $\in M$ with latency overhead ℓ and bandwidth overhead B. Let u_0, u_1, b and T be defined as in Invariant 3. If Invariant 3 is not satisfied by Π, then our adversary $\mathcal{A}_{paths}$ as in Section 4.3.2 wins.*

Proof. To prove the above, we need to prove that anonymity is broken whenever either of (i) or (ii) is false.

Whenever (i) is false, the set T is empty. Thus, the challenge message could not have been sent by the u_{1-b}.

For (ii) of the invariant to be false, both (ii.a) and (ii.b) have to be false. Note here, (ii.a) directly implies anonymity, because if one of the shares of the challenge message is dispatched by u_{1-b} within rounds $\{(r-\ell), \ldots, (r-1)\}$ there is no way for the adversary to distinguish between the challenge users.

If (ii.a) is false, (ii.b) can be false in the following ways:

1. no share of the challenge message passes through an honest node: When the adversary backtracks the paths of the shares of the challenge message starting from the recipient, the path will never cross the paths of any message from u_{1-b} at an honest node. So, $\mathcal{A}_{paths}$ can see that none of the messages from u_{1-b} is a share of the challenge message; u_{1-b} could not have sent the challenge message and hence $\mathcal{A}_{paths}$ wins.

2. At least one of the shares of the challenge message sent at $t \in T$ passes through one or more honest nodes at times t, but $\nexists\, t$ such that $t \in \{min(T), (r-1)\}$: Following the same reasoning as above, we see that paths after round $min(T)$ can be ambiguous, but there is no message from u_{1-b} before $min(T)$. So, none of them will mix with any of the shares of the challenge message. Thus, $\mathcal{A}_{paths}$ wins.

3. no message from u_{1-b} sent at $t \in T$ passes through an honest node: Similar to previous cases, when the adversary backtracks the paths of the shares of the challenge message starting

from the recipient, no path will cross the paths of the messages from u_{1-b} at an honest node. So, no message from u_{1-b} could have been a share of the challenge message.

4. At least one of the messages from u_{1-b} sent at $t \in T$ passes through one or more honest nodes at times t, but $\nexists\, t$ such that $t < r$: Following the same reasoning as above cases, we see that paths after round r can be ambiguous, but the challenge message is already delivered at round r. So, none of them will mix with any of the shares of the challenge message.

In all cases where (ii.b) is false, $\mathcal{A}_{paths}$ wins with probability 1, (assuming that (ii.a) is also false). □

Intuition about Invariant 3 and Claim 7. The invariant establishes minimal conditions for anonymity to hold against a path possibility adversary. To this end, we look at which cases would allow the adversary to defeat the protocol and in which cases the adversary can be fooled. Note that the adversary knows the two potential challenge users and can observe the traffic, but can only connect incoming and outgoing messages of a compromised party. The adversary can also see when the challenge message reaches the recipient.

- If only one of the two challenge users sends a message in the ℓ rounds before the challenge message reaches the recipient, then only that challenge user could have sent the message without violating the latency constraint. This observation is captured in part (i).

- If both challenge users happen to collaborate on sending the challenge message (u_b is the actual sender of the challenge message, u_{1-b} happens to send a share for this specific message), then the adversary cannot decide which of the two users has sent the challenge message. Even a more realistic adversary would, in most cases, lose this game. The only way to still decipher which user sent the message is to exploit other information (say, about other messages sent by the two users), but the path possibility adversary does not attempt this. This observation is captured in part (ii a).

- In case (ii a) does not occur, there are two other cases in which the path possibility adversary wins:

  (1) if the adversary can track all the shares of the challenge message from sender to recipient (since we assume (ii a) does not hold, these senders don't include u_{1-b});

(2) if the adversary can track all the packets sent by u_{1-b} to their respective recipients and thus be sure that u_{1-b} has not sent the challenge message.

We see that in both of these cases the path possibility adversary wins. Thus, they have to be necessary for anonymity. This observation is captured in part (ii b).

Overall, Invariant 3 describes the following logical formula: (i) AND (ii), where

- (ii) = (ii a) OR (ii b)
- (ii b) = (ii b 1) AND (ii b 2)

Lessons for an ideal protocol. Let us now look at the parts of Invariant 3 and discuss what they mean for constructing an ideal protocol:

- (i) The set T must not empty. This depends solely on when users send (real or noise) messages, which we capture with our definitions of the user message distribution. Thus, this part is independent of the decisions of the actual protocol.
- (ii a) The user u_{1-b} sends a share of the challenge message. To maximize this probability, which is independent of our other choices, we want the chance that any user is sending shares for any other user to be as large as possible.
- (ii b 1) At least one share of the challenge message travels through an honest node. To maximize the probability that this occurs, a protocol should maximize the number of nodes collectively visited by shares.
- (ii b 2) At least one message from u_{1-b} travels through an honest node. Since the (ideal) protocol doesn't know which users are the challenge users it needs to generalize: To maximize the probability that this occurs, the ideal protocol should maximize the number of nodes collectively visited by messages *from each user*.

These lessons, particularly (ii b), inspire our choices for an ideal protocol. Before we explore them, we briefly discuss the role of internal noise messages and relate them to the invariant.

4.3.4 Modeling internal noise

To make the accounting of bandwidth overhead easier we want to disallow the protocol from using internal noise, i.e., noise packets generated by a protocol party $\notin \mathcal{S}$. Recall the

assumptions we place on all packets, including internal noise: 1. No packets can be dropped. 2. Packets can be tagged as a share of message m, but only with one tag and that tag can never be changed. 3. Packets can remain in the system for at most ℓ rounds from their generation. 4. Shares must not violate the latency bound of the message that the noise is tagged with (c.f. Section 3.1); i.e., for a message m, all packets tagged with m must arrive within ℓ rounds of the round in which the user wanted to send m.

Claim 8 (User noise can replace internal noise)**.** *For every protocol in which noise is generated by internal protocol parties ($\notin \mathcal{S}$) and latency overhead ℓ, there exists a protocol that uses only user-generated noise (noise packets originating from a user $u \in \mathcal{S}$) and latency overhead $\ell + 1$ with at least equal probability of satisfying Invariant 3.*

Proof. We prove this claim by construction. Given a protocol Π_1 we want to construct a protocol Π_2 that satisfies the invariant with at least the same probability as Π_1. Once, an internal noise message is created, the content of the message can not be modified (although, it can be re-encrypted with different keys or decrypted), the message has to be delivered to the recipient. Additionally an internal noise message can remain in the system for $\min(\ell, z)$. where z is the latency bound for the message tag the message wants to use. Thus, having a user send a message "costs" as much as having internal nodes create the message. (Any internal noise message created not as a share of a user message will not influence the probability of the invariance being true.)

We can consider two different cases for an internal noise message:

1. **A dishonest node creates the noise message:** since, messages can not mix at a dishonest node, this does not help. Instead, a message sent by a user could help the protocol.

2. **An honest node creates the noise message:** This can definitely help the protocol. However, if a user creates the noise one round before and sends it to the given internal node in the current round, that is at least as good as a noise message created by the node in the current round.

Hence, for each internal noise message m (created at round r) in Π_1 , we make Π_2 send a noise message from a user (picked uniformly at random) at round $r - 1$. And, because of the reasons explained above, Π_2 will have at least the same probability as Π_1 in satisfying

Invariant 3. However, Π_2 now uses latency overhead $\ell + 1$ for the messages corresponding to the internal noises in Π_2 that uses latency overhead ℓ. □

Following the above claim, if a noise message is generated by an internal party P we randomly choose a user to generate it and then relay it to P.

4.3.5 Ideal protocol

We now construct a protocol Π_{ideal} that has a probability of satisfying Invariant 3 against $\mathcal{A}_{paths}$ at least as high as any other protocol in our protocol model.

Following Claim 8, we allow our ideal protocol to have latency overhead of $\hat{\ell} = \ell + 1$, and assume that every message is created by some user $u \in \mathcal{S}$. Consequently the adversary behaves as if he is dealing with a protocol that is allowed to have $\hat{\ell}$ latency overhead.

The protocol has a number of pre-defined paths. Those paths are constructed at the beginning of the protocol and do not change throughout the protocol run. Π_{ideal} has access to an oracle O (discussed later); Π_{ideal} calls O.QueryPaths() to decide the number of paths and distribution of protocol parties in each path. Fig. 4.3 presents pseudocode for the ideal protocol.

Since the protocol has control over the noise messages, it utilizes all the noise messages as shares of some real message. Whenever a message (real or noise) is sent to a path Path it is sent to the protocol party at position $r \mod |\mathsf{Path}|$ in the path, if the current round number is r. In the next round either the message is delivered to the recipient, or transferred to the next protocol party (at position $(r+1) \mod |\mathsf{Path}|$) in the same path. For every message m, Π_{ideal} queries the oracle (by calling O.QueryForMessage(m)) to decide which path the message should be sent to and the number of rounds the message should remain in the protocol. If the message is a noise message, the oracle additionally returns the real message that the noise should be a share of.

The oracle O is an overapproximation of different strategies that a protocol can use to optimize paths and noise messages. Our oracle knows the user distribution, all past and future messages, the number of compromised parties, and the protocol strategy. The protocol is oblivious to the challenge message, the challenge bit, the challenge users, the

identity of the protocol parties who are compromised; and so is the oracle. Thus, given the user distribution, the past and future messages, and the number of compromised parties, the oracle tries to maximize the probability of satisfying Invariant 3 for the protocol Π_{ideal}, against our adversary $\mathcal{A}_{paths}$.

The oracle O can achieve the above by trying out all possible configurations and calculating for each configuration the probability of satisfying Invariant 3 assuming that the two challenge users are chosen uniformly at random (refer to Fig. 4.4 for a possible instantiation of the oracle.). This consideration is different from the protocol actually satisfying the invariant, since the oracle does not actually run the protocol; is unaware of the actual challenge users, the exact protocol parties that are compromised, or the actual challenge message. Note that the oracle is not bounded polynomially anymore; however, since we are proving impossibility, a stronger protocol still provides a valid impossibility result.

```
Oracle O;
Paths ← O.QueryPaths();
MessageRoute⟨m, path, delay, tag⟩ ← empty set;

Upon Round r:
  for each Path in Paths do
    i ← r mod |Path|
    for each message m held by party Path[i − 1] do
      (path, delay, tag) ← MessageRoute.Get(m)
      if delay is expired then send m to recipient
      else send m to Path[i mod |Path|]
  for each message m in the input queue do
    (path, delay, tag) ← O.QueryForMessage(m)
    MessageRoute.Add(⟨m, path, delay, tag⟩)
    Path ← Paths[path]; send m to Path[r mod |Path|]
```

Figure 4.3. Definition of Ideal Protocol Π_{ideal}

Special case of Ideal Protocol when $\mathsf{K} > (B+1)\hat{\ell}$. For a special case of parameters, we can construct a fairly practical ideal protocol that does not require as much help from the oracle. Consider the case where the number of protocol parties K is large enough so all shares of a message can travel on distinct paths that do not overlap; also assume for simplicity that we have a constant rate of input messages per round. In this case, we can use static looping

Paths := set of paths; Delays⟨message, delay⟩;
MessagePaths⟨message, path⟩;
MessageTags⟨message, tag⟩;

Initialize(Parties P, users U, input messages I_U, protocol definition Π, latency ℓ):

PathsConfigs ← set of all possible path configuration (arrangements of parties in P)
DelaysConfigs ← Set of all possible delay (of messages) configuration of I_U
$P_{\text{global}} \leftarrow 0$
for each (PathsConfig, DelaysConfig) in (PathsConfigs, DelaysConfigs) **do**
  PathsMaps ← set of all possible mappings for messages to paths for the given DelaysConfig and PathsConfig
  TagsMaps ← all possible valid tags for messages mapping noise messages to real messages for the purpose of user coordination
  for each (PathsMap, TagsMap) ∈ (PathsMaps, TagsMaps) **do**
    $P_{\text{local}} \leftarrow$ the probability of satisfying Invariant 3 by protocol Π
    if $P_{\text{local}} > P_{global}$ **then**
      $P_{\text{global}} \leftarrow P_{\text{local}}$; Paths ← PathsConfig
      MessagePaths ← PathsMaps
      MessageTags ← TagsMap
      Delays ← DelaysConfig

QueryPaths():

return Paths

QueryForMessage(message m):

delay ← Delays.Get(m) ; tag ← MessageTags.$Get(m)$
path ← MessagePaths.Get(m)
return (path, delay, tag)

Figure 4.4. Instance of Oracle Functionality

paths, where each path is comparable to the ideal protocol from Das et al.: packets on each path remain together and hop from one node to the next (on that path).

Technically, we define $B + 1$ paths with an approximately equal number $(K/(B + 1))$ of mutually exclusive protocol parties each. Whenever a user sends a packet, the protocol queries the oracle for the latency and the path index, but the paths themselves remain the same. Otherwise the ideal protocol remains unmodified.

The ideal protocol is ideal. We now show that the ideal protocol is indeed ideal for Invariant 3.

Claim 9 (Ideal protocol is ideal for Invariant 3)**.** *Against the given adversary $\mathcal{A}_{paths}$, Π_{ideal} with latency $\hat{\ell}$ satisfies Invariant 3 with probability at least as high as any other protocol in M with latency ℓ.*

Proof. We want to prove our claim by contradiction. Suppose, there exists a protocol Π, given a latency ℓ, satisfies Invariant 3 with a higher probability than Π_{ideal} (that uses latency $\ell + 1$), against the adversary $\mathcal{A}_{paths}$. By Claim 8, we can construct a protocol Π_{new} where every message is created by some user $u \in \mathcal{S}$, and allow Π_{new} to use a latency of $\hat{\ell} = \ell + 1$; and Π_{new} will have a probability at least as much as Π_{ideal} to satisfy the invariant.

Now we construct a new protocol Π_{hybrid}, which exactly follows the strategy of Π_{ideal} with one exception: for a given message Π_{hybrid} selects the time delay t same as Π_{new}, instead of querying it from oracle O of Π_{ideal}.

The ideal strategy for ensuring that at least one honest party is on at least one the path of the messages from u_{1-b} is to ensure that as many distinct parties as possible are on all the paths combined. Similarly, the possibility of having an honest party of the paths of the shares of the challenge message is also maximized by maximizing the number of distinct parties on all those paths combined.

Similarly, the ideal strategy for obfuscating the challenge sender with user coordination is by maximizing the number of users sending shares for the challenge message. Since the user distribution is the same for both Π_{new} and Π_{hybrid}, Π_{hybrid} is at least as successful in satisfying the invariant due to the oracle.

For both Π_{new} and Π_{hybrid}, the times when messages are sent and the time delays are same, and hence, for every message the path length is same for both Π_{new} and Π_{hybrid}. However, Π_{hybrid} decides the number of paths, and distribution of the protocol parties on those paths by querying the oracle. Hence, Π_{hybrid} has a probability of satisfying Invariant 3 at least as high as Π_{new}.

Now, if we compare Π_{hybrid} and Π_{ideal} : they follow the same strategy. But Π_{ideal} picks the time delay t for any message from oracle O such that t is *optimal*. Hence, Π_{ideal} satisfies Invariant 3 with probability at least as high as Π_{hybrid}. Thus, Π_{new} does not satisfy Invariant 3 with a higher probability than Π_{ideal}. □

From here onwards, we assume that messages (real or noise) are generated only by users $\in \mathcal{S}$, and whenever a latency of ℓ is allowed to the protocol, we allow the ideal protocol to have a latency of $\hat{\ell} = \ell + 1$ in our calculations.

4.4 Analyzing synchronized users

The first user distribution we analyze is the synchronized user message distribution U_B as defined in Section 6.2. Recall that in U_B in every round exactly one user sends a message and within N rounds each user sends a message only once. We allow the protocol to add up to B noise packets per round and leave it up to the protocol to decide which users send those B packets. If B is not a natural number, we allow the protocol to send $\lfloor B \rfloor$ noise messages per round and one more every few rounds such that the average bandwidth overhead remains B while spacing them out as evenly as possible.

4.4.1 Lower bound on adversarial advantage

Theorem 4.4.1. *For user distribution U_B, no protocol $\Pi \in M$ can provide δ-sender anonymity, for any*

$\delta < \left(1 - \frac{B}{\mathsf{N}-1}\right)\left[1 - \frac{(\tau+1)\mathsf{N} - B\hat{\ell} - \hat{\ell}}{\mathsf{N}} g(\tau) - \frac{B\hat{\ell}+\hat{\ell}-\tau\mathsf{N}}{\mathsf{N}} g(\tau+1)\right]$

where $\tau = \lfloor \frac{B\hat{\ell}+\hat{\ell}}{\mathsf{N}} \rfloor$, $\hat{\ell} = \ell + 1$

and $g(x) = \begin{cases} 1 & \mathsf{c} < x\hat{\ell} \\ 1 - \binom{\mathsf{c}}{x\hat{\ell}} \Big/ \binom{\mathsf{K}}{x\hat{\ell}} & \mathsf{c} \geq x\hat{\ell}. \end{cases}$

Proof. Suppose u_0 and u_1 are the challenge users, and u_b sends the challenge message which reaches the recipient in some round r. We know from Claim 9 that Π_{ideal} is ideal; thus, we focus on Π_{ideal} here. By definition of Π_{ideal}, the challenge message can have up to $(B+1)$ shares, including the one sent by u_b. Since the challenge users are not known to the oracle O, the best strategy for O is to have B shares per real message.

For our invariant to be satisfied, it is necessary that u_{1-b} sends at least one message within $[r-\ell, r-1]$. Such a message can be a share of the challenge message, or a real message. If we denote by x the number of messages sent by u_{1-b} in a given interval of $\hat{\ell}$ rounds, x can

have only two possible values depending on the values of B, $\hat{\ell}$ and N. That is because the protocol tries to maximize the total number of users that send messages in a given interval of ℓ rounds. Hence, u_{1-b} sends $\tau = \lfloor \frac{B\hat{\ell}+\hat{\ell}}{\mathsf{N}} \rfloor$ messages with probability $\frac{(\tau+1)\mathsf{N}-B\hat{\ell}-\hat{\ell}}{\mathsf{N}}$, and sends $(\tau+1)$ messages with probability $\frac{B\hat{\ell}+\hat{\ell}-\tau\mathsf{N}}{\mathsf{N}}$.

If none of them is a share of the challenge message, we require that at least one of those messages passes through an honest node before round r. Hence,

$$
\begin{aligned}
&\Pr\left[\text{Invariant 3 is true}\right] \\
&\leq \Pr\left[u_{1-b} \text{ sends a share of the challenge message}\right] \\
&+ \Pr[u_{1-b} \text{ sends no shares of the challenge message} \\
&\qquad \wedge\, u_{1-b} \text{ sends a message in the given span of } \hat{\ell} \text{ rounds}] \\
&\qquad \times \Pr\left[\text{At least one of the messages visits an honest node}\right] \\
&\leq \frac{B}{\mathsf{N}-1} + \left(1-\frac{B}{\mathsf{N}-1}\right)\frac{(\tau+1)\mathsf{N}-B\hat{\ell}-\hat{\ell}}{\mathsf{N}} \times g(\tau) \\
&\qquad + \left(1-\frac{B}{\mathsf{N}-1}\right)\frac{B\hat{\ell}+\hat{\ell}-\tau\mathsf{N}}{\mathsf{N}} \times g(\tau+1).
\end{aligned}
$$

where $\tau = \lfloor \frac{B\hat{\ell}+\hat{\ell}}{\mathsf{N}} \rfloor$, and $g(x)$ is a function that provides an upper bound on the probability that at least one message from u_{1-b} passes through at least one honest node in a given interval of $\hat{\ell}$ rounds, when u_{1-b} sends exactly x messages. Hence,

$$
\begin{aligned}
&\Pr[\text{at least one message from } u_{1-b} \text{ passes through} \\
&\qquad \text{an honest node } | u_{1-b} \text{ sends } x \text{ messages}] \\
&\leq g(x) = \begin{cases} 1 & \mathsf{c} < x\hat{\ell} \\ 1 - \binom{\mathsf{c}}{x\hat{\ell}} \Big/ \binom{\mathsf{K}}{x\hat{\ell}} & \mathsf{c} \geq x\hat{\ell} \end{cases}
\end{aligned}
$$

By Claim 7 whenever Invariant 3 is not true the adversary wins. Whenever it is true, the adversary still can flip a coin and thus the probability that the adversary loses is bounded by the following:

$$
\begin{aligned}
\Pr\left[0 = \mathcal{A}_{paths} | b = 1\right] &= \Pr\left[1 = \mathcal{A}_{paths} | b = 0\right] \\
&\leq \frac{1}{2}\Pr\left[\text{Invariant 3 is true}\right].
\end{aligned}
$$

Therefore,

$$\begin{aligned}\delta &\geq 1 - \Pr\left[\text{Invariant 3 is true}\right] \\ &\geq \left(1 - \frac{B}{\mathsf{N}-1}\right)\left[1 - \frac{(\tau+1)\mathsf{N} - B\hat{\ell} - \hat{\ell}}{\mathsf{N}} g(\tau) - \frac{B\ell + \ell - \tau\mathsf{N}}{\mathsf{N}} g(\tau+1)\right].\end{aligned}$$

□

Although the above bound is a perfectly valid lower bound for δ over $0 \leq \mathsf{c} \leq \mathsf{K}$, when $\mathsf{c} < \hat{\ell}$ and $\tau = 0$, we can derive a more precise lower bound on δ:

$$\delta \geq \left(1 - \tfrac{B}{\mathsf{N}-1}\right)\left(1 - \tfrac{B(\hat{\ell}-\mathsf{c})+(\hat{\ell}-\mathsf{c})}{\mathsf{N}} - \tfrac{B\mathsf{c}+\mathsf{c}}{\mathsf{N}}\left[1 - 1\Big/\binom{\mathsf{K}}{\mathsf{c}}\right]\right).$$

The following section presents a derivation for this bound.

4.4.2 A tighter special case for $\mathsf{c} < \hat{\ell}$

When $\tau = 0$ and $\mathsf{c} < \hat{\ell}$, we can derive a more precise bound than the one in Theorem 4.4.1. Since $\tau = 0$, there is at most one message sent by u_{1-b} in a span of $\hat{\ell}$ rounds. There is a chance that u_{1-b} does not send a message, the invariants are not satisfied (and the adversary wins) in that case. When u_{1-b} sends a message, the invariants are satisfied only if the whole path of the message is not compromised. However, since $\mathsf{c} < \hat{\ell}$, the adversary can not compromise a whole path of length $\hat{\ell}$. Therefore, the adversary has a chance to break the invariants if the message from u_{1-b} is dispatched in $\{r - \mathsf{c}, \ldots, r-1\}$. If the message is sent by u_{1-b} in $\{r - \hat{\ell}, r - \mathsf{c} - 1\}$, the invariants can be satisfied. Therefore, we can derive a lower bound on δ as follows:

$$\begin{aligned}\delta \geq\ & \Pr[u_{1-b} \text{ does not send a share of challenge message}] \\ & \times \Big(1 - \Pr[u_{1-b} \text{ sends a message in } \{r - \hat{\ell}, r - \mathsf{c} - 1\}] \\ & \quad - \Pr[u_{1-b} \text{ sends a message in } \{r - \mathsf{c}, r - 1\}] \\ & \quad \times \Pr[\text{At least one of the } \mathsf{c} \text{ parties is honest}]\Big) \\ \geq\ & \left(1 - \frac{B}{\mathsf{N}-1}\right)\left(1 - \frac{B(\hat{\ell}-\mathsf{c}) + (\hat{\ell}-\mathsf{c})}{\mathsf{N}} - \frac{B\mathsf{c}+\mathsf{c}}{\mathsf{N}} \times \left[1 - 1\Big/\binom{\mathsf{K}}{\mathsf{c}}\right]\right)\end{aligned}$$

Bound on anonymity when $B\hat{\ell} \leq \mathsf{N}$ and $\mathsf{c} < \hat{\ell}$. We can use a similar technique as above to derive a precise bound on δ when $B\hat{\ell} \leq 1$ and $\mathsf{c} < \hat{\ell}$. Since $B\hat{\ell} \leq \mathsf{N}$, for $\hat{\ell} < N$ the number

of messages sent by bob is bounded by 2, and $\tau \leq 1$. Therefore, we can derive the following lower bound on δ:

$$\begin{aligned}
\delta \geq\ & \Pr[u_{1-b} \text{ does not send a share of challenge message}] \\
& \times \Big(1 - \Pr[u_{1-b} \text{ sends two messages in } \{r-\hat{\ell}, r-1\}] \\
& - \Pr[u_{1-b} \text{ sends only one message in } \{r-\hat{\ell}, r-\mathsf{c}-1\}] \\
& - \Pr[u_{1-b} \text{ sends only one message in } \{r-\mathsf{c}, r-1\}] \\
& \quad \times \Pr[\text{At least one of the } \mathsf{c} \text{ parties is honest}]\Big) \\
\geq\ & \left(1 - \tfrac{B}{\mathsf{N}-1}\right)\left(1 - max\left(0, \tfrac{B\hat{\ell}+\hat{\ell}-\mathsf{N}}{\mathsf{N}}\right)\right. \\
& \left. - \tfrac{B(\hat{\ell}-\mathsf{c})+(\hat{\ell}-\mathsf{c})}{\mathsf{N}} - \tfrac{B\mathsf{c}+\mathsf{c}}{\mathsf{N}} \times \left[1 - 1\Big/\binom{\mathsf{K}}{\mathsf{c}}\right]\right)
\end{aligned}$$

Note that, Pr[At least one of the c parties is honest] can never be negligible. Because, even for $c = 1$, $1\Big/\binom{\mathsf{K}}{\mathsf{c}}$ is not negligible. The adversary can always choose to compromise less number of parties if that gives the adversary more advantage. This untightness is because of the approximations in our proof, tighter bounds are left for future work.

Therefore, a protocol can not achieve strong anonymity if $max\left(0, \frac{B\hat{\ell}+\hat{\ell}-\mathsf{N}}{\mathsf{N}}\right) + \frac{B(\hat{\ell}-\mathsf{c})+(\hat{\ell}-\mathsf{c})}{\mathsf{N}}$ is not overwhelmingly 1.

4.4.3 Impossibility for strong anonymity

Using Theorem 4.4.1, we can derive the following impossibility theorems for AC protocols to achieve strong anonymity.

Theorem 4.4.2. *For user distribution U_B with $K, N \in poly(\eta)$, $K > c$, $\hat{\ell} < \mathsf{N}$, $\mathsf{N} - 1 > B \geq 0$, no protocol $\Pi \in M$ can achieve strong anonymity if*

(i) $\hat{\ell}(B+1) < \mathsf{N} - \epsilon(\eta)$ *where $\epsilon(\eta) = 1/\eta^d$ for a positive constant d, OR*

(ii) $c \geq (B+1)\hat{\ell}$ *and* $\hat{\ell} \in O(1)$.

Theorem 4.4.2 in words. For our user distribution U_B in which user behavior is synchronized and somewhat predictable, a protocol has only three options to achieve strong anonymity: (1) to use a massive amount of latency overhead $\hat{\ell} \geq \mathsf{N}$ (intuitively: "wait until everyone has sent a message"); (2) to use a massive amount of bandwidth overhead $B \geq \mathsf{N}-1$

(intuitively: "every user sends a packet in every round"); or (3) to have a trade-off between the two: $\hat{\ell}(B+1) \geq N$ (intuitively: "make sure that messages wait long enough for everyone to have sent a packet"). In case (3) we have an additional requirement: if the adversary is allowed to compromise more than the number of parties that any message and its shares can meet while they are within the protocol, then it is possible that the challenge message and all its shares only travel through compromised nodes. If the length of the paths taken by these messages is constant, this occurs with non-negligible probability and thus strong anonymity is impossible.

Proof of Theorem 4.4.2. We know,
$\delta \geq \left(1 - \frac{B}{\mathsf{N}-1}\right)\left[1 - \frac{(\tau+1)\mathsf{N} - B\hat{\ell} - \hat{\ell}}{\mathsf{N}} g(\tau) - \frac{B\ell + \ell - \tau\mathsf{N}}{\mathsf{N}} g(\tau+1)\right]$.

First, we observe that, if $B\hat{\ell} + \hat{\ell} < \mathsf{N} - \frac{1}{\eta^x}$, τ is zero, and hence, $g(\tau)$ is zero. Moreover, $\frac{B\hat{\ell}-\hat{\ell}}{\mathsf{N}} < 1 - \frac{1}{\mathsf{N}\eta^x}$ = not overwhelming. Which means δ cannot be negligible. Now,

$$B\hat{\ell} + \hat{\ell} < \mathsf{N} - \epsilon(\eta) \iff (B+1)\hat{\ell} < \mathsf{N} - \epsilon(\eta).$$

We additionally need both $g(\tau)$ and $g(\tau+1)$ to be overwhelming to achieve strong anonymity. When $c \geq (B+1)\hat{\ell}$, both $\tau(\hat{\ell}+1)$ and $(\tau+1)\hat{\ell}$ have to be in $\omega(1)$ (i.e., not in $O(1)$), in order for $g(\tau)$ and $g(\tau+1)$ to become overwhelming. We know that $B < \mathsf{N}-1 \implies \frac{B}{\mathsf{N}} < 1$. If $\hat{\ell}$ is in $O(1)$,
$\tau = \lfloor \frac{B\hat{\ell}+\hat{\ell}}{\mathsf{N}} \rfloor = \lfloor \left(\frac{B}{\mathsf{N}}\hat{\ell} + \frac{\hat{\ell}}{\mathsf{N}}\right) \rfloor \leq (\ell+1) \in O(1)$.
Hence, $\tau\hat{\ell}$ is also in $O(1)$. Therefore, $g(\tau)$ and $g(\tau+1)$ are not overwhelming. □

Theorem 4.4.3 (Anytrust Impossibility Theorem)**.** *For user distribution U_B with $\mathsf{K}, \mathsf{N} \in$ poly(η), $\mathsf{K} - \mathsf{c} = \gamma \in O(1)$, $\mathsf{K} > \mathsf{c}$, no protocol $\Pi \in M$ can achieve strong anonymity if $\hat{\ell} \leq \mathsf{N} - 1$ and $B \leq \mathsf{N} - 2$ and $\hat{\ell}^2 \leq \mathsf{K} - \gamma$.*

Theorem 4.4.3 in words. If all but a constant number of nodes are compromised, then strong anonymity is only possible with a massive latency overhead or a massive bandwidth overhead (cases (1) and (2) from Theorem 4.4.2 in words respectively) or if the latency overhead ℓ allows each packet to traverse at least the square root of all compromised parties ($\hat{\ell}^2 \geq \mathsf{c}$).

Proof of Theorem 4.4.3. We know, $\tau = \lfloor \frac{B\hat{\ell}+\hat{\ell}}{\mathsf{N}} \rfloor < \hat{\ell}$. When $\hat{\ell}^2 < \mathsf{K} - \gamma$,

$$\frac{\binom{\mathsf{c}}{\tau\hat{\ell}}}{\binom{\mathsf{K}}{\tau\hat{\ell}}} \geq \frac{\binom{\mathsf{c}}{\hat{\ell}^2}}{\binom{\mathsf{K}}{\hat{\ell}^2}} \geq \frac{\mathsf{c}!\,(\mathsf{K}-\hat{\ell}^2)!}{(\mathsf{c}-\hat{\ell}^2)!\,\mathsf{K}!} \geq \frac{(\mathsf{K}-\gamma)!\,\gamma!}{\mathsf{K}!}$$

For $\gamma \in O(1)$, the above quantity is always non-negligible. Hence, $g(\tau)$ is never overwhelming. Therefore, δ cannot be negligible unless $\hat{\ell} \geq \mathsf{N} - negl(\eta)$ or $B \geq \mathsf{N} - 1 - negl(\eta)$. □

4.5 Analyzing unsynchronized users

We now analyze the unsynchronized user message distribution U_P as defined in Section 6.2. Recall that in U_P in every round each user tosses a biased coin with success probability $p \in (0, 1]$ to decide whether or not to send a message. This coin toss is independent of coins tossed by other users or in other rounds. We assume that the bandwidth overhead is part of p, i.e., we divide up p into the probability to send a real message $p < p$ and define our bandwidth overhead as $B = \frac{p-p}{p}$ noise messages per real message.

4.5.1 Lower bound on adversarial advantage

Theorem 4.5.1. *For user distribution U_P, no protocol $\Pi \in M$ can provide δ-sender anonymity, for any [1ex]*

$$\delta < \left(1 - \frac{B_{\text{eff}}}{\mathsf{N}-1}\right)\left[1 - g(\hat{\ell}) \times f_p^{SA}(\hat{\ell})\right], \text{ where}$$

$$B_{\text{eff}} = \min(B, \hat{\ell}p\mathsf{N} - 1),$$

$$f_p^{SA}(d) = \min\left(1, \frac{1}{2} + \left(1 - (1-p)^d\right)\right), \text{ and}$$

$$g(x) = \begin{cases} 1 - \binom{\mathsf{c}}{x\hat{\ell}} / \binom{\mathsf{K}}{x\hat{\ell}} & x\hat{\ell} \leq \mathsf{c} \leq \mathsf{K} \\ 1 & \text{otherwise.} \end{cases}$$

Proof of Theorem 4.5.1. Suppose u_0 and u_1 are challenge users, and u_b sends the challenge message. The challenge reaches the recipient at round r. The challenge message can have up to $B = \frac{p-p}{p}$ additional shares (excluding the share sent by u_b). Ideally, we want u_{1-b} to send at least one of the $\frac{p-p}{p}$ shares. If not, we at least want u_{1-b} to send at least one message in $[r - \hat{\ell}, r - 1]$, that passes through an honest node before round r.

By Invariant 3, only the shares sent in rounds $\{(r-\hat{\ell}),\ldots,(r-1)\}$ can contribute to anonymity. Therefore, the number of shares for the challenge message is bounded by $B_{\text{eff}} = min(B, \hat{\ell}p\mathsf{N}-1)$.

The probability that u_{1-b} sends at least one message within a span of $\hat{\ell}$ rounds is upper bounded by $f_p^{SA}(\ell)$ as explained in Section A.1.1. Moreover, u_{1-b} can not send more than $\hat{\ell}$ messages in $\hat{\ell}$ rounds. Thus, we can derive:

$$
\begin{aligned}
&\Pr\left[\text{Invariant 3 is true}\right] \\
&\leq \Pr\left[u_{1-b} \text{ sends a share of the challenge message.}\right] \\
&\quad + Pr[u_{1-b} \text{ does not send a share of the challenge message} \\
&\quad \wedge\, u_{1-b} \text{ sends a message in the given span of round } \hat{\ell}] \\
&\quad \times Pr[\text{Some share of the challenge message visits honest node} \\
&\qquad \text{and some message from } u_{1-b} \text{ visits honest node}] \\
&= \frac{B_{\text{eff}}}{\mathsf{N}-1} + \left(1-\frac{B_{\text{eff}}}{\mathsf{N}-1}\right) \\
&\times \left(\sum_{d=0}^{Z} Pr\left[\text{At least one node in d paths is honest}\middle|\ X^{(u_{1-b})}=d\right] \times \Pr\left[X^{(u_{1-b})}=d\right]\right. \\
&\quad \left. + \Pr\left[\text{At least one node in Z paths is honest}\right] \times \Pr\left[X^{(u_{1-b})}>Z\right]\right) \\
&\leq \frac{B_{\text{eff}}}{\mathsf{N}-1} + \left(1-\frac{B_{\text{eff}}}{\mathsf{N}-1}\right) \times Pr\left[\text{At least one node is honest in Z paths}\right] \\
&\quad \times \Pr\left[u_{1-b} \text{ sends at least one message in} \{(r-\hat{\ell}),\ldots,(r-1)\}\right] \\
&\leq \frac{B_{\text{eff}}}{\mathsf{N}-1} + \left(1-\frac{B_{\text{eff}}}{\mathsf{N}-1}\right) \times g(Z) \times \left(1-(1-p)^{\hat{\ell}}\right)
\end{aligned}
$$

By Claim 7, whenever Invariant 3 is not satisfied the adversary wins, bounding the adversary's advantage by:

$$
\begin{aligned}
\delta &\geq 1 - \Pr\left[\text{Invariant 3 is true}\right] \\
&\geq \left(1-\frac{B_{\text{eff}}}{\mathsf{N}-1}\right)\left[1-g(\hat{\ell}) \times f_p^{SA}(\hat{\ell})\right].
\end{aligned}
$$

□

Although the above bound is a valid bound for $0 \leq \mathsf{c} \leq \mathsf{K}$, we can derive a more precise bound when $\mathsf{c} < \ell$:

$$
\begin{aligned}
\delta \geq &\left(1-\tfrac{B_{\text{eff}}}{\mathsf{N}-1}\right) \times \left(1-f_p^{SA}(\hat{\ell}-\mathsf{c})\right) \\
&\times \left[1-f_p^{SA}\left(\mathsf{c}\right)\left(w_2+w_1\left[1-1/\binom{\mathsf{K}}{\mathsf{c}}\right]\right)\right],
\end{aligned}
$$

where $w_1 = \mathsf{c}p\,(1-p)^{\mathsf{c}-1}$ and $w_2 = 1 - w_1 - (1-p)^{\mathsf{c}}$. We refer to Section 4.5.3 for the derivation of this bound.

4.5.2 Impossibility for strong anonymity

Using Theorem 4.5.1, we can derive the following impossibility theorems.

Theorem 4.5.2. *For user distribution U_P, with $\hat{\ell} < \mathsf{N}$ and $B < (\mathsf{N}-1) - \epsilon(\eta)$, no protocol $\Pi \in M$ can achieve strong anonymity if $p\hat{\ell} < 1 - \epsilon(\eta)$. Moreover, strong anonymity can not be achieved if $\hat{\ell} \in O(1)$.*

Theorem 4.5.2 in words. We confirm that a protocol even with user coordination generally can only provide strong anonymity if it (1) uses a massive amount of bandwidth overhead $B \geq \mathsf{N} - 1$ (intuitively: "for every real message, every other user sends a share"); or (2) satisfies the bound for U_P without compromised nodes from Chapter 3. Thus, we confirm that for $B < \mathsf{N} - 1$ their basic trilemma condition (without compromised nodes) holds even against protocols with user coordination. In other words, while user coordination with $B < \mathsf{N}$ strengthens a protocol against compromised parties, it does not suffice for overcoming the basic trilemma condition.

Proof of Theorem 4.5.2. If $B < (\mathsf{N}-1) - \epsilon(\eta)$, $\frac{B}{\mathsf{N}-1}$ will be less than $1 - neg(\eta)$. Hence, $H = \left(1 - g(\hat{\ell}) \times f_p^{SA}(\hat{\ell})\right)$ has to be negligible to achieve strong anonymity. However, this is a generic lower bound on δ, and from Section A.1.1 we know that it is sufficient to consider $\left(1-(1-p)^{\hat{\ell}}\right)$ instead of $f_p^{SA}(\hat{\ell})$. Hence, we require $H = \left(1 - g(\hat{\ell}) \times \left(1-(1-p)^{\hat{\ell}}\right)\right)$ to be negligible for the protocol to achieve strong anonymity. When $p\hat{\ell} < 1-\epsilon(\eta) \implies (1-p)^{\hat{\ell}} < 1-\epsilon(\eta)$, H can never be negligible, and consequently, δ can never be negligible.

Even when $(1-p)^{\hat{\ell}}$ is negligible, $g(\hat{\ell})$ has to be overwhelming as well to achieve strong anonymity in case $\mathsf{c} > \ell$, which implies $\left[\binom{\mathsf{c}}{\hat{\ell}^2} \Big/ \binom{\mathsf{K}}{\hat{\ell}^2}\right]$ has to be negligible (since $\mathsf{c} \geq \hat{\ell}^2 \implies \mathsf{c} \geq \hat{\ell}^2$), to achieve strong anonymity. $\binom{\mathsf{c}}{\hat{\ell}^2}/\binom{\mathsf{K}}{\hat{\ell}^2}$ can never be negligible if $\hat{\ell}^2 \in O(1)$.

When $\mathsf{c} < \hat{\ell}$, We need $\left[1 - 1/\binom{\mathsf{K}}{\mathsf{c}}\right]$ to be overwhelming to achieve strong anonymity. This means, we need the term $1/\binom{\mathsf{K}}{\mathsf{c}}$ to be negligible and that never happens for a constant c. If

$\hat{\ell} \in O(1)$, c is also $O(1)$, since $\hat{\ell} > \mathsf{c}$ by our assumption. And, that shows that our theorem holds for $\mathsf{c} < \hat{\ell}$ as well.

Finally consider the case $\hat{\ell} \leq \mathsf{c} < \hat{\ell}^2$. For a constant ℓ, if a constant c can provide adversary better advantage, the adversary will choose to compromise fewer protocol parties even though he can compromise more. Therefore, Whenever we have constant $\hat{\ell}$, it is impossible to achieve strong anonymity, since it is impossible even for c being as small as 1. □

Theorem 4.5.3. *For user distribution U_P, Given $p < 1 - \epsilon(\eta)$, $B < (\mathsf{N} - 1) - \epsilon(\eta)$ $\frac{\mathsf{c}}{\mathsf{K}} = const$, no protocol $\Pi \in M$ can achieve strong anonymity if $c > \hat{\ell}^2$ and $\hat{\ell}^2 \in O(log(\eta))$, where $\epsilon(\eta) = 1/\eta^x$ for a positive constant x.*

Theorem 4.5.3 in words. If a constant fraction $\frac{\mathsf{c}}{\mathsf{K}}$ of protocol parties is compromised and the protocol does not use a massive bandwidth overhead (see above), then the latency has to grow significantly with the security parameter ($\hat{\ell}$ must grow superlogarithmic in η).

Proof of Theorem 4.5.3. We have $\delta \geq \left(1 - \frac{B_{\text{eff}}}{\mathsf{N}-1}\right)\left[1 - g(\hat{\ell}) \times f_p^{SA}(\hat{\ell})\right]$. If $B < \mathsf{N} - 1 - \epsilon(\eta)$, $\left(1 - \frac{B_{\text{eff}}}{\mathsf{N}-1}\right)$ cannot be negligible. In that case, both $f_p^{SA}(\hat{\ell})$ and $g(\hat{\ell}) = 1 - \binom{\mathsf{c}}{\hat{\ell}^2}/\binom{\mathsf{K}}{\hat{\ell}^2}$ have to be overwhelming to make δ negligible. For $\mathsf{c} > \hat{\ell}^2$ and $\frac{\mathsf{c}}{\mathsf{K}} = const = \frac{1}{y}$,

$$\frac{\mathsf{c} - \hat{\ell}^2}{\mathsf{K} - \hat{\ell}^2} > \frac{1}{y} \iff \left(\frac{\mathsf{c} - \hat{\ell}^2}{\mathsf{K} - \hat{\ell}^2}\right)^{\hat{\ell}^2} > \left(\frac{1}{y}\right)^{\hat{\ell}^2}$$

$$\implies \frac{\mathsf{c} \ldots (\mathsf{c} - \hat{\ell}^2)}{\mathsf{K} \ldots (\mathsf{K} - \hat{\ell}^2)} > \left(\frac{1}{y}\right)^{\hat{\ell}^2}$$

$\left(\frac{1}{y}\right)^{\hat{\ell}^2}$ cannot be negligible for $\hat{\ell}^2 \in O(log(\eta))$. □

Theorem 4.5.4. *For user distribution U_P, given $B < (\mathsf{N} - 1) - \epsilon(\eta)$, no protocol $\Pi \in M$ can achieve strong anonymity if $p \times \max\left\{\hat{\ell} - \mathsf{c}, \frac{\hat{\ell}}{2}\right\} < 1 - \epsilon(\eta)$.*

This shows that if the adversary compromises more protocol parties p has to grow accordingly to provide strong anonymity. Not only that, $\hat{\ell}$ needs to grow as c grows.

Theorem 4.5.4 in words. For U_P, if the protocol does not use a massive bandwidth overhead (see above), then compromised parties reduce the effective latency in terms of the basic trilemma by a factor of up to two; the more parties can be compromised, the harder it becomes for the protocol.

Proof of Theorem 4.5.4. When $B < (\mathsf{N}-1) - \epsilon(\eta)$, $\left(1 - \frac{B_{\text{eff}}}{\mathsf{N}-1}\right)$ can never be negligible. Additionally, because $p(\hat{\ell} - \mathsf{c}) < 1 - \epsilon(\eta)$, $(1-p)^{\hat{\ell}-\mathsf{c}}$ can not be negligible. Therefore, to achieve strong anonymity, $(1-(1-p)^{\mathsf{c}})$ and $\left[1 - 1/\binom{\mathsf{K}}{\mathsf{c}}\right]$ has to be overwhelming – that is not possible if $p\mathsf{c} < 1 - \epsilon(\eta)$. [Here we use the knowledge from Section A.1.1 to use $(1-p)^{\hat{\ell}-\mathsf{c}}$ instead of $f_p^{SA}(\hat{\ell} - \mathsf{c})$ and $(1-(1-p)^{\mathsf{c}})$ instead of $f_p^{SA}(\mathsf{c})$.]

Finally, note that the adversary can always choose to compromise less than c nodes and thus would choose to compromise $\frac{\hat{\ell}}{2}$ at most to maximize the advantage.

□

Theorem 4.5.5 (Anytrust Impossibility Theorem)**.** *For user distribution U_P with $\mathsf{K}, \mathsf{N} \in$ poly(η), $\mathsf{K} - \mathsf{c} = \gamma \in O(1)$, $\mathsf{K} > \mathsf{c}$, no protocol $\Pi \in M$ can achieve strong anonymity if $\hat{\ell} \leq \mathsf{N} - 1$ or $B \leq \mathsf{N} - 2$ or $\hat{\ell}^2 \leq \mathsf{K} - \gamma$.*

This theorem is similar to Theorem 4.4.3 from Section 4.4. If there are only constant number of honest nodes, strong anonymity is impossible without a large latency overhead or a huge bandwidth overhead or if the latency overhead $\hat{\ell}$ allows each packet to traverse at least $\sqrt{\mathsf{c}}$ parties. The proof is also similar to that of Theorem 4.4.3, therefore we skip the proof here.

4.5.3 A tighter special case for $\mathsf{c} < \ell$

Let us derive a tighter upper bound on δ, in case of unsynchronized user message distribution, when $0 \leq \mathsf{c} < \ell$. W is a random variable denoting the minimum number of paths that the adversary needs to compromise to ensure no honest party on the paths of the shares of the challenge messages as well as no honest party on the paths of the messages from u_{1-b}. When $\mathsf{c} < \hat{\ell}$,

$$
\begin{aligned}
&\Pr\left[\text{Invariant 3 is true}\right] \\
\leq &\Pr\left[u_{1-b} \text{ sends a share of the challenge message.}\right] \\
&+ Pr[u_{1-b} \text{ does not send a share of the challenge message}] \\
&\wedge \Big(Pr[u_{1-b} \text{ sends a message in } \{r-\hat{\ell}, r-\mathsf{c}-1\}] \\
&+ Pr[u_{1-b} \text{ does not send a message in } \{r-\hat{\ell}, r-\mathsf{c}-1\}] \\
&\times \Big(Pr[u_{1-b} \text{ sends more than one message in } \{r-\mathsf{c}, r-1\}] \\
&+ Pr[u_{1-b} \text{ sends only one message in } \{r-\mathsf{c}, r-1\}] \\
&\times Pr[\text{Some share of the challenge message visits honest node} \\
&\quad \text{and some message from } u_{1-b} \text{ visits honest node}] \Big) \Big) \\
\leq &\frac{B_{\text{eff}}}{\mathsf{N}-1} + \left(1-\frac{B_{\text{eff}}}{\mathsf{N}-1}\right)\Bigg[\left(1-(1-p)^{\hat{\ell}-\mathsf{c}}\right) + (1-p)^{\hat{\ell}-\mathsf{c}} \\
&\times \Big(\Pr\left[W \geq 1 \wedge X^{u_{1-b}}(c) \geq 2\right] \\
&+ \Pr\left[W = 0 \wedge X^{u_{1-b}}(c) \geq 1\right] \times \left[1 - 1/\binom{\mathsf{K}}{\mathsf{c}}\right] \Big)\Bigg] \\
\leq &\frac{B_{\text{eff}}}{\mathsf{N}-1} + \left(1-\frac{B_{\text{eff}}}{\mathsf{N}-1}\right)\Bigg[\left(1-(1-p)^{\hat{\ell}-\mathsf{c}}\right) + (1-p)^{\hat{\ell}-\mathsf{c}} \\
&\times \Big(\Pr\left[W \geq 1 \wedge X^{u_{1-b}}(c) \geq 1\right] \\
&+ \Pr\left[W = 0 \wedge X^{u_{1-b}}(c) \geq 1\right] \times \left[1 - 1/\binom{\mathsf{K}}{\mathsf{c}}\right] \Big)\Bigg] \\
\leq &\frac{B_{\text{eff}}}{\mathsf{N}-1} + \left(1-\frac{B_{\text{eff}}}{\mathsf{N}-1}\right)\Bigg[\left(1-(1-p)^{\hat{\ell}-\mathsf{c}}\right) + (1-p)^{\hat{\ell}-\mathsf{c}} \\
&\times \Pr\left[X^{u_{1-b}}(c) \geq 1\right]\left(\Pr\left[W \geq 1\right] + \Pr\left[W = 0\right]\left[1 - 1/\binom{\mathsf{K}}{\mathsf{c}}\right]\right)\Bigg] \\
\leq &\frac{B_{\text{eff}}}{\mathsf{N}-1} + \left(1-\frac{B_{\text{eff}}}{\mathsf{N}-1}\right)\Bigg[\left(1-(1-p)^{\hat{\ell}-\mathsf{c}}\right) + (1-p)^{\hat{\ell}-\mathsf{c}} \\
&\times (1-(1-p)^{\mathsf{c}})\left(\Pr\left[W \geq 1\right] + \Pr\left[W = 0\right]\left[1 - 1/\binom{\mathsf{K}}{\mathsf{c}}\right]\right)\Bigg]
\end{aligned}
$$

Note that W is a random variable, where $W = min\left(\frac{(X-X)}{X}+1, X\right)$. Here X and X follow $Binom(\mathsf{c}, p)$ and $Binom(\mathsf{c}, p)$ respectively. Therefore, We can say that $Pr[W=1]$ is bounded by $Pr[W=1] \leq w_1 = Pr[X=1] = \mathsf{c}p\,(1-p)^{\mathsf{c}-1}$. Consequently, $Pr[W>1] \geq w_2 = Pr[X>1] = 1 - w_1 - (1-p)^{\mathsf{c}}$. Therefore, we can write,

$$
\begin{aligned}
&\Pr\left[\text{Invariant 3 is true}\right] \\
&\leq \frac{B_{\text{eff}}}{\mathsf{N}-1} + \left(1 - \frac{B_{\text{eff}}}{\mathsf{N}-1}\right)\left[f_p^{SA}(\hat{\ell}-\mathsf{c}) + \left(1 - f_p^{SA}(\hat{\ell}-\mathsf{c})\right)\right. \\
&\quad \left.\times f_p^{SA}(\mathsf{c})\left(w_2 + w_1\left[1 - 1/\binom{\mathsf{K}}{\mathsf{c}}\right]\right)\right] \\
&\text{Thus,} \\
&\delta \geq 1 - \Pr\left[\text{Invariant 3 is true}\right] \\
&\geq 1 - \tfrac{B_{\text{eff}}}{\mathsf{N}-1} - \left(1 - \tfrac{B_{\text{eff}}}{\mathsf{N}-1}\right)\left[f_p^{SA}\left(\hat{\ell}-\mathsf{c}\right) + \left(1 - f_p^{SA}(\hat{\ell}-\mathsf{c})\right)\right. \\
&\left.\times f_p^{SA}\left(\mathsf{c}\right)\left(\Pr\left[W \geq 1\right] + \Pr\left[W = 0\right]\left[1 - 1/\binom{\mathsf{K}}{\mathsf{c}}\right]\right)\right] \\
&\geq \left(1 - \tfrac{B_{\text{eff}}}{\mathsf{N}-1}\right) \times \left[1 - f_p^{SA}\left(\hat{\ell}-\mathsf{c}\right) - \left(1 - f_p^{SA}(\hat{\ell}-\mathsf{c})\right)\right. \\
&\left.\times \left(1-(1-p)^{\mathsf{c}}\right)\left(\Pr\left[W \geq 1\right] + \Pr\left[W = 0\right]\left[1 - 1/\binom{\mathsf{K}}{\mathsf{c}}\right]\right)\right] \\
&\geq \left(1 - \tfrac{B_{\text{eff}}}{\mathsf{N}-1}\right) \times \left(1 f_p^{SA}(\hat{\ell}-\mathsf{c})\right)\left[1 - f_p^{SA}\left(\mathsf{c}\right)\right. \\
&\quad \left.\times \left(\Pr\left[W \geq 1\right] + \Pr\left[W = 0\right]\left[1 - 1/\binom{\mathsf{K}}{\mathsf{c}}\right]\right)\right]
\end{aligned}
$$

4.6 Discussion of results

4.6.1 Impossibility results

From our impossibility theorems in Sections 4.4 and 4.5, we observe that strong anonymity requires a combination of latency overhead and bandwidth overhead – which is very similar to our observations Chapter 3. The strong assumption of user coordination (U.C.) appears to reduce the cost to achieve anonymity, but in most cases does not suffice. As an exception, strong anonymity can always be achieved with user coordination for $B \geq \mathsf{N}$ – even in cases where it is provably impossible for protocols that do not use user coordination.

In Table 4.1 we compare the impossibility results for protocols with user coordination with those for protocols without user coordination. We compare different cases for the number of compromised nodes c in relation to the latency overhear ℓ and the bandwidth overhead B. Each entry shows under which condition we can prove that strong anonymity is impossible. Recall that the impossibility bounds cannot be tight, as we solely consider the possible paths adversary $\mathcal{A}_{paths}$. Tight bounds would have to also make requirements about

the message distributions. However, the results are comparable to those of Chapter 3, since we use the same adversary $\mathcal{A}_{paths}$ in both the cases.

Table 4.1. Impossibility Conditions for Anonymous Communication, with number of protocol-nodes K, number of compromised protocol parties c, number of clients N, latency overhead ℓ. In all cases we assume that $\ell < \mathsf{N}$, $1 \leq B < (\mathsf{N}-1) - \epsilon(\eta)$, and $\epsilon(\eta) = 1/\eta^d$ for a positive constant d. We compare different cases for the number of compromised nodes c in relation to the latency overhead ℓ and the bandwidth overhead B. Each entry shows under which condition we can prove that strong anonymity is impossible. Note that we allow protocols with U.C. to utilize a latency of $\hat{\ell} = \ell + 1$ (c.f., Footnote 3).

Cases	U_B without U.C.	U_B, with U.C.	U_P without U.C.	U_P, with U.C.
$\mathsf{c} \geq 0$	$\ell(B+1) < \mathsf{N} - \epsilon$	$\hat{\ell}(B+1) < \mathsf{N} - \epsilon$	$\ell p < 1 - \epsilon$	$\hat{\ell} p < 1 - \epsilon$
$0 < \mathsf{c} \leq \ell$	$\frac{(\ell - \mathsf{c})(B+1)}{(\mathsf{N} - \epsilon)} < 1$	$\frac{(\hat{\ell} - \mathsf{c})(B+1)}{(\mathsf{N} - \epsilon)} < 1$	$(\ell - \mathsf{c})p < 1 - \epsilon$	$p(\hat{\ell} - \mathsf{c}) < 1 - \epsilon$
$\ell < \mathsf{c} \leq B\ell + \ell$	$\ell \in O(1)$	$\hat{\ell}(B+1) < \mathsf{N} - \epsilon$	$\ell \in O(1)$	$\hat{\ell} \in O(1)$
$(B+1)\ell < \mathsf{c}$	$\ell \in O(1)$	$\hat{\ell} \in O(1)$	$\ell \in O(1)$	$\hat{\ell} \in O(1)$
$\mathsf{K}/\mathsf{c} \in O(1)$	$\ell \in \log(\eta)$	$\hat{\ell} \in \sqrt{\log(\eta)}$	$\ell \in \log(\eta)$	$\hat{\ell} \in \sqrt{\log(\eta)}$

Unified impossibility bound for both user distributions. When comparing our impossibility results for both user distributions, we can represent them with a single unified impossibility condition $\hat{\ell}(p + \beta) < 1 - \epsilon(\eta)$, where β is the number of noise messages per user per round. For the unsynchronized user message distribution, $\beta = pB = p - p$. For the synchronized user distribution, $\beta = \frac{B}{\mathsf{N}} = pB$, since p by definition is $\frac{1}{\mathsf{N}}$.

Limitations of our results. In our derivations we do not consider a probabilistic adversary which indeed has a higher chance of deanonymizing users. Additionally, we do not count the cost of user coordination in our results. These factors make our results untight, still giving us a strict lower bound on the cost of anonymity in terms of latency and bandwidth overhead.

We also assume that no user ever goes offline, which means that any restrictions we prove in our protocol model directly translate to both protocols that have an *always online* representation of users and protocols that are more vulnerable. In other words: strong anonymity might be even harder to achieve in practice. This makes our analysis slightly more untight for protocols that don't provide solutions for coping with offline users and set intersection attacks.

Conversely, notions weaker than "strong anonymity", e.g., a partial but robust anonymity set, can be easier to achieve. However, if the cardinality of such a partial set is known in

advance our analysis can be easily adapted by reducing the "set of all users" to the partial set and then following our methodology to compute bounds: If N_W is the cardinality of an anonymity set W, our bounds will hold for parameter N_W instead of N.

Table 4.2. Interesting cases for AC, with number of protocol-nodes K, number of compromised protocol parties c, number of clients N, latency overhead ℓ. The table assumes for all rows $\mathsf{N} \in \Theta(\eta^2)$, $\mathsf{K} \in \Theta(\eta)$, $\ell < \mathsf{K} < \mathsf{N}$ and $B \leq (\mathsf{N}-2)$. Here, ✗ denotes that strong anonymity is provably impossible and (✓) denotes that we could not show this impossibility, i.e., strong anonymity could be possible. In some cases the impossibility proofs rely on additional requirements, i.e., we can only show ✗ if these requirements are met. Note that we allow protocols with U.C. to utilize a latency of $\hat{\ell} = \ell + 1$ (c.f., Footnote 3).

	U_B without U.C.		U_B with U.C.		U_P without U.C.		U_P with U.C.	
Interesting cases	Ano.	Add.req.	Ano.	Add.req.	Ano.	Add.req.	Ano.	Add.req.
$\beta\ell = 1, \ell < \mathsf{K}$, $\mathsf{c} \in \Theta(\log(\mathsf{K}))$	✗		✗		✗	$p < \frac{1}{\ell}$	✗	$p < \frac{1}{\ell}$
$\beta = \frac{1}{\ell}, \ell \in O(1)$, $\mathsf{K} - \mathsf{c} \in \Theta(\eta)$	✗		✗		✗		✗	
$\beta = \frac{1}{\ell}, \ell < \mathsf{c} < \ell^2$, $\mathsf{K} - \mathsf{c} \in \Theta(1)$	✗		✗		✗		(✓)	
$\beta = \frac{1}{\ell}, \ell^2 \leq \mathsf{c}$, $\mathsf{K} - \mathsf{c} \in \Theta(1)$	✗		✗		✗		✗	
$\beta = \frac{1}{\sqrt{\ell}}, \ell^2 \leq \mathsf{c}$, $\mathsf{K} - \mathsf{c} \in \Theta(1)$	✗		✗		✗		✗	
$\beta\ell \in O(1), \mathsf{c} = \frac{\mathsf{K}}{2}$, $\ell \leq \log(\mathsf{K})$	✗		✗		✗	$p < \frac{1}{2}$	✗	$p < \frac{1}{2}$
$\beta > \frac{1}{\log(\eta)}, \mathsf{c} = \frac{\mathsf{K}}{4}$, $\ell \geq \log(\mathsf{K})$	✗		(✓)		✗		(✓)	
$\ell < \frac{\log(\eta)}{2}$, $\mathsf{c} = \mathsf{K} - 1$	✗		✗		✗		✗	

4.6.2 Interesting cases & corner cases

This section discusses some boundary cases and some interesting cases to breathe life into our necessary constraints. We discuss combinations of bandwidth overhead B, latency overhead ℓ, and number c of compromised nodes with respect to the impact of utilizing user coordination (U.C.) in an ACN. In Table 4.1 we compare the impossibility results for those cases for protocols with user coordination with those cases. Here, ✗ denotes that strong anonymity is provably impossible and (✓) denotes that we could not show this impossibility, i.e., strong anonymity could be possible. In some cases the impossibility proofs rely on additional requirements, i.e., we can only show ✗ if these requirements are met.

Our results are dominated by the universal necessary constraints without any compromisation, i.e., $\hat{\ell}(p+\beta) < 1-\epsilon(\eta)$. Hence, the focus of Table 4.2 is to show which combinations of parameters along the lines $\hat{\ell}(p+\beta) = 1$ are impossible for which scenario. We illustrate that, while U.C. might lead to strongly anonymous ACNs in some cases, there are interesting cases along the lines of the universal necessary constraints where even ACNs with U.C. cannot achieve strong anonymity.

When we compare the results for protocols without user coordination vs. protocols with user coordination, we compare $\ell = x$ vs. $\hat{\ell} = x$ to induce fairness[3]. For deciding the verdicts, we directly use the lower bounds on δ from our results.

Constant latency, full bandwidth overhead. Let us consider ACNs that send for every real message N shares ($B = N$), which we call full bandwidth overhead. In this case, from our lower bounds on δ we can observe that U.C. has an impact, as no internal node is needed to achieve strong anonymity, as is done in DC-nets [12]. As a consequence, even if there are internal parties but all internal parties are compromised U.C. leaves the possibility of achieving strong anonymity (e.g., along the lines of DC-nets). Without U.C., strong anonymity is impossible if the latency is short ($\ell \in O(1)$). However, when a protocol does not have full bandwidth overhead, U.C. can not provide strong anonymity without the help of latency overhead and honest intermediate parties.

Almost very high latency, high bandwidth overhead. For high latency bounds $\ell \leq \mathsf{K} - 1$ that are just shy of visiting every node in the ACN ($\ell = \mathsf{K}$), strong anonymity is impossible for synchronized users, even if a high amount of bandwidth overhead $B = \mathsf{N}/\ell$ or $\beta = 1/\ell$ is tolerated. (In Table 4.2 we use β to unify the impossibility bounds for synchronized and unsynchronized user message distribution, where $\beta = pB$; for synchronized users, always $p = 1/\mathsf{N}$.) In Section 4.4.2 we provide additional calculations relevant for these corner cases. For the unsynchronized user distribution, strong anonymity is impossible if the rate p at which real messages are sent per round is low, roughly $p < 1/\ell$.[4]

[3]↑When we allow latency to be $\ell + 1$ for protocols with user coordination to approximate noise generated by internal parties with user noise, we also allow protocols with only user noise to have latency $\ell + 1$. It is unfair to compare them with protocols without user coordination with latency ℓ. Moreover, when $\ell = 0$, there is no intermediate party, so there is no internal noise.

[4]↑Recall that we here assume that $\mathsf{N} \in \theta(\eta^2)$ and $\mathsf{K} \in \theta(\eta)$.

Moderate latency, minimal bandwidth overhead. Next, we consider interesting cases where we fix the latency ℓ and consider a bandwidth overhead in such a way that β is along the lines of $\beta\ell = 1$. For the synchronized user distribution, if the latency $\ell \approx \sqrt{\eta} \approx \sqrt{\mathsf{K}}$ and $B = \mathsf{N}/\ell$, our results leave the possibility for strong anonymity only if the total number of compromised parties is less than ℓ, i.e., $\ell > \mathsf{c}$. For the unsynchronized user distribution, for similar latency ($\ell \approx \sqrt{\mathsf{K}}$) and compromisation up to $\mathsf{c} \leq \ell^2$, strong anonymity is possible and the bandwidth overhead can be as low as $B = \beta/p = O(1)$ for a high rate of real messages (p is a constant fraction). If all nodes but one are compromised ($\mathsf{c} = \mathsf{K}-1$), strong anonymity is impossible for both user distributions when $\ell < \sqrt{\mathsf{K}}$, independent of the bandwidth overhead — which confirms our Anytrust impossibility theorem (Theorems 4.4.3 and 4.5.5).

Log latency, with nearly full bandwidth overhead. Along the line $\beta\ell = 1$, another interesting case is $\ell = \log(\mathsf{K})/2$. In this case, the latency overhead is so low that there is no chance to evade a pervasive adversary that compromises a lot of nodes ($\mathsf{c} \geq \mathsf{K}/2$). In a more specific case, strong anonymity is impossible in a strong compromisation scenario where all nodes but one are compromised ($\mathsf{c} = \mathsf{K} - 1$), regardless of the bandwidth overhead, i.e., for any $\beta < 1 - \epsilon(\eta)$ and $B \leq (\mathsf{N} - 2)$. For a slightly higher latency $\ell \geq 2\log(\mathsf{K})$ and a weak adversary with $\mathsf{c} \leq \mathsf{K}/4$, we cannot exclude the possibility for strong anonymity as long as the universal necessary constraints are satisfied ($\hat{\ell}(p + \beta) \geq 1$).

4.7 Implications and scope

Our novel necessary constraints for the core of ACNs with user coordination describe a large set of lower bounds for combinations of bandwidth overhead, latency overhead, resistance to compromised parties, and the degree of anonymity. The rich literature on ACNs contains a few proposals that come close to these novel necessary constraints. This section discusses some of these ACNs, in particular, their usage of user coordination to achieve stronger anonymity and the user coordination online overhead against passive adversaries, i.e., without the overhead of DoS countermeasures.

In this chapter, to keep the presentation concise, we mainly discuss protocols that utilize user coordination. For protocols that do nor utilize user coordination, we refer to the

Table 4.3. Latency vs. bandwidth vs. strong anonymity of AC protocols, with the number of protocol-nodes K, number of clients N, and message-threshold T, expected latency ℓ per node, dummy-message rate β.

Protocol	**Latency**	**Bandwidth**	**Strong Anonymity**
Tor [13]	$\theta(1)$	$\theta(1/\mathsf{N})$	impossible
Hornet [14]	$\theta(1)$	$\theta(1/\mathsf{N})$	impossible
Herd [7]	$\theta(1)$	$\theta(\mathsf{N}/\mathsf{N})$	possible
Riposte [6]	$\theta(N)$	$\theta(\mathsf{N}/\mathsf{N})$	possible
Vuvuzula [2]	$\theta(\mathsf{K})$	$\theta(\mathsf{N}/\mathsf{N})$	possible
Riffle [8]	$\theta(\mathsf{K})$	$\theta(\mathsf{N}/\mathsf{N})$	possible
Threshold mix [30]	$\theta(T\mathsf{K})$	$\theta(1/\mathsf{N})$	impossible*
Loopix [1]	$\theta(\sqrt{K}\ell)$	$\theta(\beta)$	possible
DC-Net [11, 12]	$\theta(1)$	$\theta(\mathsf{N}/\mathsf{N})$	possible
Dissent-AT [5]	$\theta(1)$	$\theta(\mathsf{N}/\mathsf{N})$	possible
DiceMix [15]	$\theta(1)$	$\theta(\mathsf{N}/\mathsf{N})$	possible

* if T in $o(poly(\eta))$

discussion in Chapter 3. However, to provide a totalistic perception of our results, we plot both types (one that utilizes user coordination and one that does not) of protocols on a 2D graph (see Figure 4.5); our constraints mark an area on the graph where strong anonymity is impossible with latency overhead (x-axis) versus bandwidth overhead (y-axis). Additionally, in Table 4.3 we summarize the bounds on the bandwidth β and latency overhead ℓ of different protocols (with and without user coordination).

Chaum started a line of work on so-called DC-nets [5, 10–12] that implements anonymous broadcast channels assuming users have agreed on some cryptographic keys with each other or with the protocols parties and have decided on a schedule to ensure that only one real message is sent in each round. As we model the single-recipient setting and assume a passive adversary, for communication overhead analyses we assume variants where broadcast implementations are replaced by directed messages from protocol parties to the dedicated recipient. In particular, for [11, 12], in every round, we assume that each party sends either a real message or a noise message (a share in our model) to the recipient. As a protocol in our model, each client would in each round send packets with the same tag and one of these packets would contain the real message, leading to a bandwidth overhead of N. With

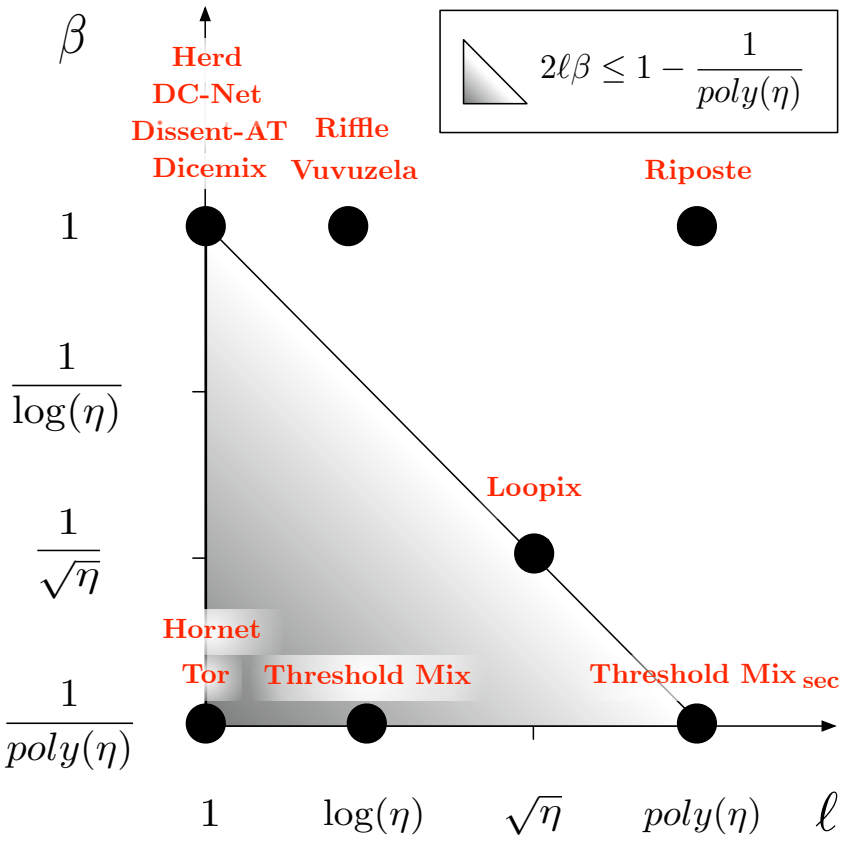


Figure 4.5. Asymptotic latency overhead (ℓ) and bandwidth overhead (β) together with the "area of impossibility" where $2\ell\beta \leq 1 - \epsilon(\eta)$. We portray protocols as dots depending on their choices for ℓ and β. Technically, if we use Theorem 3.4.2, we β is replaced by $p = \beta + p$, where p is the rate at which users send messages. This graph assumes N is ca. $poly(\eta)$, the number of nodes K is ca. $\log \eta$. The threshold for Threshold Mix $T = 1$ and for Threshold Mix_{sec} $T = \mathsf{N} = poly(\eta)$. In the graph, both the axes are approximately in logarithmic scale. (For a more accurate visual representation we refer the readers to Chapter A.3 and the trilemma website [52].)

$B = N$ and $\ell = 0$, DC-nets satisfy our novel necessary constraints for ACNs with user coordination from Theorem 4.5.3, thereby showing tightness of our bounds for this border case. Concerning the complexity of the user coordinations, Chaum proposed [12] a solution where two messages sent at the same time over the broadcast channel would collide. To avoid collisions, he proposed to divide the broadcast into N^2 blocks and to constantly maintain a separate reservation array of size N^2 in the broadcast that is sent in each round. Even if only 1-bit messages are sent, this protocol results in an additional bandwidth overhead of $2N^2$ and one additional round. This bandwidth overhead can also be spread over the latency by spreading the reservation array of the blocks over several rounds.

Herbivore [10] partitions the set of clients into several subsets of size N/q (for some integer $q > 1$) and solely implements DC-nets within a partition, effectively reducing the bandwidth B to the size N/q of a partition. With $B = N/q$ and $\ell = 1$, our results prove that Herbivore cannot achieve the employed AnoA-styled notion of strong sender anonymity, which is easy to see: if the two challenge senders u_0, u_1 are from different partitions, an adversary can easily win. Herbivore also uses the concept of reservations for avoiding collisions, yet also provides several bandwidth-latency sweet spots.

Dissent-AT [5] also reduces DC-nets communication overhead. It relies on K computation servers (the K protocol parties in our model). Assuming that every client has a shared secret with each of the K servers, each client only has to send her real message or share to one of the K servers. Afterwards, in our model, these K servers send their combined shares to the dedicated recipient. Hence, the bandwidth overhead is N messages for each real message, except that these N messages are not sent to N parties as in DC-nets (leading to a communication overhead of N^2) but only to one of the K servers (leading to a communication overhead of N). As we assume a single recipient, in our comparison the bandwidth overhead is $B = \mathsf{N}$ just as for DC-nets. Hence, Dissent-AT satisfies our necessary constraints for ACNs with user coordination from Theorem 4.5.3; so, our results do not exclude strong anonymity for Dissent-AT. DISSENT-AT uses a verifiable shuffle among the K servers and results in a periodic latency overhead of K.

Dicemix [15] is outside the scope of our model (see Section 4.1.1), as it can mix shares with different tags, yet it nevertheless obeys our bounds. Dicemix aims at removing the scheduling

requirements of other DC-nets. Dicemix assumes that each party sends a message, and in our synchronized user distribution, it has to wait for N rounds until real messages arrive. The protocol requires 4 communication rounds[5] leading to a latency of $N + 4$ in our model, which includes the user coordination's collision-avoidance subprotocol. Every party sends N packets whenever all messages have been collected (in every Nth round); so, the bandwidth overhead (per client) in our model is $B = N$. If we average the overhead ($B = B/N = 1$) over N rounds, however, Dicemix is close to our universal bound $N = B\ell \geq N$, hence our results do not rule out strong anonymity, even if almost all other parties are compromised.

All of the above protocols only deal with a boundary condition from our results and their bandwidth overheads are tremendous. To the best of our knowledge, none of the ACNs with user coordination utilize the combination of multi-hop layered encryption feature (as used in mix-nets) with user coordination features that render the real sender's packet indistinguishable from a noise message, even for the recipients. Indeed, there is significant scope for improvement here specially if we need to reduce the bandwidth overhead by introducing some latency overhead.

ACNs with global static synchronization (i.e., U_B with U.C.) effectively introduce large overhead (e.g., N rounds for DC-nets), since each user has to wait for its turn to send a message. Hence, such ACNs are difficult to use with low-latency applications. Moreover, current designs with user coordination (e.g., DC-nets or Dissent-AT) can only then provide a full anonymity[6] set (encompassing all clients) in the Anytrust setting ($\mathsf{c} = \mathsf{K} - 1$) if almost all clients send a dummy message (i.e., $B = \mathsf{N} - 1$). For dynamic user coordination (U_P with U.C.), our results, however, do not exclude strong anonymity for $B < \mathsf{N}$ if sufficient latency is added, as in the third row of Table 4.2: $B = \beta/p = 1/(\ell p) = q/\sqrt{\mathsf{K}}$ (for $p = 1/q$ and a constant q), $\ell = \sqrt{\mathsf{K}}, \mathsf{c} = \mathsf{K} - 1, \mathsf{K} = \eta, \mathsf{N} = \mathsf{K}^2$. Such overhead combinations might be interesting for future exploration of ACN designs.

[5]↑While Dicemix includes integrity protection and self-healing mechanism that leads to $4+2f$ communication rounds for one message if f peers are deviating from the protocol, these mechanisms do not kick in if all peers follow the protocol (as even the compromised parties do in our analysis), leading to only 4 communication rounds.

[6]↑Satisfying strong anonymity implies achieving a full anonymity set.

One possible direction is to reconsider the recent mix-net protocol designs [1, 2, 7, 53] in light of user coordination. In particular, our lower bounds indicate that these designs could benefit from incorporating user coordination techniques, which could increase their resistance against compromisation (by increasing the bandwidth overhead B) while reducing latency overhead ℓ. Another possibility, for employing the user coordination, is to consider Riposte design [6], which uses the private information storage primitive. In Riposte, to enable the recipient to point to the exact incoming packet a sender input needs to include a number of elements proportional to the square root of the size of the whole stored database. Using user coordination can allow the Riposte-like design to reduce this bandwidth overhead by sending a smaller number of elements.

5. IMPLICATIONS OF TRILEMMA RESULTS: CONSTRUCTING AC PROTOCOL AT THE COST OF LATENCY OVERHEAD

In the last two chapters we analyzed the fundamental constraints of AC protocols. In this chapter and the next we are going to use those results and construct efficient AC protocols. In the process we aim the identify the fundamental building blocks that are required to achieve anonymity.

Starting with Chaum [4], over the last thirty years, numerous anonymous communication (AC) protocols have been designed and some implemented. Among those, million of users today employ the Tor network for their privacy over the Internet. However, while aiming for low communication and latency overhead, the Tor network [49] employs onion routing [54] and it demonstrated to be significantly vulnerable to traffic analysis empirically [17, 20, 55] as well as conceptually [24].

Many other older and newer AC protocols offer improved anonymity against traffic analysis by introducing delays and incorporating dummy messages. With a growing demand for better network anonymity, especially in the blockchain world, a few of them are getting actively developed for real-world use.[1] While many AC protocol offers improved protection against traffic analysis w.r.t. Tor and several of those demonstrate their practicality with performance analysis, the offered anonymity guarantees are often not formally well-defined, restricted to a particular application and pre-processing, or out-rightly broken.

Dining cryptographers' networks (DC-nets) [12] and its successors [11, 15, 35] offer easily provable sender anonymity, low latency and can be scaled up to a certain extent via the any-trust kind of assumption on the servers; however, these protocols inherently demand to fix the users participating in a protocol-round in advance, and expect those users to agree on pair-wise symmetric keys and cannot manage any churn. Mixing network (mixnet) type protocols like Yodel [56], Karaoke [31], Stadium [33], and Vuvuzela [2] can manage the user churn while offering a differential-privacy kind of guarantee. They also can scale to thousands of users, which maintaining the on-path latency delay to be less than a minute; however, the users need

[1]↑e.g.: nymproject, https://xx.network/

to send messages with those at the beginning of every instance.[2] Moreover, these protocols come with a special requirement of pre-agreement between clients about who is communicates with whom using some kind of dialing protocol. Finally, differential privacy kind of security degrades very fast with the compositions, and even sightly improving the privacy guarantee (by increasing ϵ value) can introduce significant communication overhead. Atom [32] offers an indistinguishability kind of anonymity guarantee instead while maintaining similar user support via horizontal scaling; however, in the process, it does introduce a significant latency delay and results in the delay of around 28 minutes.

Recently, an interesting line of research has emerging that leverages (interactive or non-interactive) two-party or multi-party computation as a service towards offering anonymity [6, 53, 57–59]. These protocols are inherently compute-bound and cannot scale well as the number of MPC-parties and the number of users increase [58, 59]. Moreover, several of those come with rather rigid system architecture, communication flow and adversarial assumptions [53]. While recent MPC-based protocols have improved in terms of scalability [6, 57], they are only scalable when deployed with a very low number (e.g., two or three) nodes (i.e., computation servers), resulting in a semi-centralized trust model. In particular, relying on very few nodes does not enable a truly decentralized ACN where the trust is widely distributed, as in MixNets.

The recent Loopix [1] system comes up with a mixnet design offering a tunable knob between latency overhead and the required traffic volume to offering protection against traffic analysis. Here, the users keep on sending messages at intervals following a probability distribution independent of all other users. Loopix tries to offer anonymity using randomized/probabilistic delay for messages at each routing node. However, Loopix does not provide *end-to-end* provable anonymity guarantees.

Therefore, in the quest for a good AC protocol the following question remains: Can we avoid the restricted communication flows of Karaoke, Stadium, and Atom or the rigid architectural requirements of anonymity via MPC-as-a-service or ad-hoc delays of Loopix, while still offering provable anonymity in a scalable fashion?

[2]↑In the layman terms, once missed your (ferry) boat, you cannot join in between; you have to wait for the boat to return to the starting position.

We answer those important questions in this chapter by presenting a mixnet-type protocol that does not require the users to follow a restricted communication pattern, does not introduce ad-hoc delays on the routing nodes, and solely relies on the mixing on messages in the nodes to achieve anonymity.

In this chapter we are going to present a mixnet-type protocol *Streams* that realizes similar security properties as a trusted third party stop-and-go mix while allowing a fraction of mixnodes to be compromised by the adversary, with a latency overhead of several seconds (depending on the proportion of compromised nodes). As long as each message stays in the system for at least the given amount of time before being delivered, our protocol provides the same security guarantees comparable to that of a trusted third party.

An important feature of this protocol is that it can scale for a hundred thousand clients using a novel construction that we call *supernodes*, while realizing the above security property and keeping the latency overhead under several seconds (for a fraction of 20% compromised nodes, the end-to-end latency remains under 8 seconds).

5.1 System Goals and Protocol Overview

We consider a typical mix network based architecture [1, 32] allowing users to broadcast messages anonymously using an infrastructure of mix nodes. Our main objective is to design a scalable anonymous communication protocol that provides pairwise unlinkability, a generalization of *tail indistinguishability* [60] and unlinkabilty [61]. We relate pairwise unlinkability to sender anonymity and relationship anonymity in Section 5.6.3.

In this section, we offer an overview of the system model, the protocol idea and the desired properties of the protocol, and then show that the protocol provides pairwise unlinkability against global passive adversaries with full access to network traffic.

5.1.1 System Model and Security Goals

Similar to provable-secure mix-net systems such as [32], we assume that protocols parties works in time epoch and protocol rounds. All clients and nodes synchronize their rounds such that there will be a designated entry (protocol) node for every round and the previous

designated node and the clients are suppose to send their packets to this nodes before the round starts.

Being consistent with the protocol model used in the previous chapters, we consider a global passive adversary with full access to network traffic. Additionally, the adversary can passively compromise some users as well as some non-user protocol parties. Those passively compromised protocol parties are considered *honest-but-curious*: they still follow the protocol description but attempt to extract information.

As a basic measure for anonymity, we focus on *pairwise unlinkability* in the context of this protocol, i.e., the (in-) ability of a third party to figure out which among two messages entering a system corresponds to which of the two same messages leaving the system at a later point. This notion of unlinkability is closely related to other prominent anonymity notions, such as sender anonymity (which of two potential senders has sent a specific message?) and relationship anonymity (which people are in communication with each other?).

Definition 5.1.1 (Pairwise unlinkability)**.** *A protocol provides* pairwise unlinkability *of messages over time t up to probability δ for $0 \leq \delta < 1$ if any pair of messages $(u_0, m_0, t_{s,0}, t_{f,0}, R_0)$ and $(u_1, m_1, t_{s,1}, t_{f,1}, R_1)$, where u is the sender of the message, m the content, t_s the time the message enters the system, t_f the time the message leaves the system, and R the receiver of the message, with $\min t_{f,0}, t_{f,1} - \max t_{s,0}, t_{s,1} \geq t$ cannot be distinguished from the pair $(u_1, m_0, t_{s,1}, t_{f,0}, R_0)$ and $(u_0, m_1, t_{s,0}, t_{f,1}, R_1)$ with an advantage greater than δ.*

Informally, we say that the two messages *are shuffled* from the adversary's point of view. We refer to Section 5.6.3 for a discussion on how pairwise unlinkability relates to sender anonymity and relationship anonymity.

Even though the provable anonymity analysis focuses on the global passive adversaries, the system already incorporates integrity protection using the standard cryptographic methods [62]. Moreover, following Loopix [1], we present updates for our protocol to defend against active attackers in Section 5.6.1.

Non-goals. Finally, in this work we focus on an application independent protocol and do not consider side-channel attacks — the detailed analyses of fingerprinting of web-browsing

and other side-channel that might arise in specific application scenarios are out of scope for this work.

5.1.2 Protocol Idea

Our protocol Streams is inspired by the *ideal protocol* in the recent anonymity lower-bound analysis [24], a hypothetical protocol shown to be ideal at confusing a simplified adversary.

This ideal protocol, used to show limitations of anonymous communication, makes use of an oracle to determine many of the important decisions. In Streams, we approximate the oracle to achieve a slightly weaker version of anonymity — we only achieve pairwise mixing instead of sender anonymity. In Section 5.6.3 we discuss more about the leakage our protocol has, and we conjecture that the leakage cannot be avoided by any real protocol (even a trusted third party stop-and-go mix) unless a large amount of bandwidth overhead is introduced; however, we do not yet have a proof for that impossibility.

We now describe below the key ideas behind the protocol and its inherent limitations.

Assumptions. Our anonymity proofs assume that users are online and actively participate in the protocol. We leverage the *Sphinx* packet format that provides end-to-end encryption for all messages, and a node does not learn the path length and the relay position of the node on the path of a packet.

Protocol strategy. Influenced by the ideal protocol presented in Chapter 3, we design Streams that tries to keep all messages in the system together across different rounds, which allow us offers end-to-end provable anonymity efficiently (as compared to the state of the art). In particular, in a given round, all clients and all other nodes funnel all their messages (except the messages that need to be delivered in that round) to a single designated node. While the system may very-well create a circular list of all node and keep following that, we prefer to change it using randomness beacons (refer to Fig. 5.1) so that the adversary cannot strategically compromise nodes towards increasing its advantage.

Our aim is also make the system scale to millions of messages. Using one designated node for a round can become bottleneck there: for million users, it will have to perform a

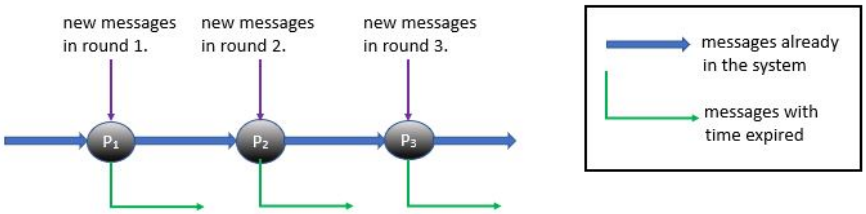


Figure 5.1. Routing Strategy in Streams

millions of public key cryptographic operations; this can significantly slow down the system as the round duration will be at least a few second if not minutes.

We overcome this problem using a novel *supernode* structure. A supernode consist of a set of compute nodes that perform cryptographic operations and a funnel node that collects and shuffles all decrypted messages. Streams routes messages through a series of supernodes, so that each message in the system alternates between compute nodes and funnel nodes. Funnel nodes present a common path shared by all messages; leaving a funnel node, the stream of messages fans out, spanning potentially many different compute nodes, before merging again in the next funnel node, thus allowing us to scale up the system considerably while providing strong pairwise unlinkability. Pairwise unlinkability is achieved if messages are jointly kept in the system for long enough to meet and mix.

Randomness Beacons and Path synchronization. We assume randomness beacons [63, 64] available to all the protocol parties. A randomness beacon [65] emits a new *random* value at intermittent intervals such that the emitted values are bias-resistant, i.e., no entity can influence a future beacon value, and unpredictable, i.e., no entity can predict future beacon value. NIST's Randomness Beacons project [63] and the emerging Drand Organization [64] are two prominent ready to use Internet-based instantiations of randomness beacon, while several other protocols [66–70] and implementations [71–73] are also available. In Streams, at any given round, every protocol party can use the messages from the beacons to derive the next ℓ nodes of the common path.

5.1.3 Desired Properties of the Protocol

Below we summarize the key considerations behind the design of the protocol Streams:

1. Streams provides strong pairwise unlinkability of messages, only requiring a modest latency overhead: if the latency is in $\omega(\log \eta)$, the probability that two messages are pairwise unlinkable is overwhelming in η. We formally prove our claims about Streams's pairwise unlinkability in Section 5.4.
2. Streams is designed for scalability and can manage up to a million messages in the system at any given time.
3. The flexibility to choose the parameter ℓ that is appropriate for a given application scenario.
4. Streams can be hardened against active attacks; we provide guidance towards that in Section 5.6.1.

5.2 Protocol Description

Here we present the system setup, the protocol design of Streams, and how our *supernode* design helps the protocol scale for a large number of users.

5.2.1 System Setup

We consider a set $\mathcal{S}$ of users communicating to a set $\mathcal{R}$ of recipients through a set I of intermediate nodes (or just 'nodes'). In real life, the same user can act as sender as well as recipient, however, we consider the sender role and recipient role as two separate logical entities. Each sender is denoted by u_i where $i \in \{1, \ldots, \mathsf{N}\}$ and $|\mathcal{S}| = \mathsf{N}$. Similarly, each recipient is denoted by R_i where $i \in \{1, \ldots, \mathsf{N}\}$ and $|\mathcal{R}| = \mathsf{N}$.

We consider global passive adversaries that can statically compromise up to c nodes out of a total of $\mathsf{K} = |I|$ nodes. In this section, we only consider passive compromisation, which means that the compromised protocol parties still follow the protocol specifications, however the adversary has access to all the internal states of a compromised party. In Section 5.6.1 we discuss the necessary adaptations for the protocol against an active and adaptive adversary.

Our protocol uses a round-based communication model and synchronized clocks. In Section 5.6.2, we discuss how our results can be extended to loosely synchronized clocks.

We consider the availability of a public key infrastructure (PKI) to all the users and nodes. For each party (client or node) P there exist a private public key pair $(\mathtt{sk}_P, \mathtt{pk}_P)$. For a party P to send a message to a party Q, P needs to know the public key $\mathtt{pk}_Q$ of Q. We assume such a PKI system can be instantiated using a one time setup similar to [1, 14, 49, 54, 74] — the exact procedure is out of scope for this work. The PKI is only used by the onion subprotocol Π_{sub} that we use from the work of Kuhn et al. [60]. We summarize all the system parameters in Figure 3.1.

5.2.2 Model

We use a hybrid world UC model [75] to represent our protocol – where the protocol has access to some additional ideal (hybrid) functionalities that is available to the protocol as well as the adversary. A protocol party (an honest user or node) or the adversary can access such a functionality through an incorruptible ITI $\mathcal{F}$ that provides certain ideal guarantees, e.g., clock time, common reference string (CRF), key registration etc. More specifically, our formalization uses four functionalities: a round-based communication functionality $\mathcal{F}_{round}$, a globally available randomness beacon $\mathcal{F}_{CRF}$, a key registration functionality $\mathcal{F}_{RKR}$, and a secure communications sessions functionality $\mathcal{F}_{SCS}$. The environment $\mathcal{E}$ can access those ideal functionalities either through the protocol parties or through the adversary.

We consider static compromisation by the adversary $\mathcal{A}$, which means the adversary can corrupt protocol parties before the environment starts the protocol run. Whenever a corrupted party is activated, $\mathcal{A}$ is run instead of the protocol code. We discuss how to defend against slow dynamic corruption in Section 5.6.1.

Round Functionality $\mathcal{F}_{round}$. Streams is a round-based protocol. To model that all messages are sent in rounds, we introduce a hybrid functionality $\mathcal{F}_{round}$ (see Figure 5.2) to enforce rounds on the protocol parties. We ensure that the environment $\mathcal{E}$ activates the honest parties in every round. $\mathcal{F}_{round}$ ensure, though, that the environment $\mathcal{E}$ cannot activate a protocol party multiple times in the same round by keeping track of the $Rounds[i]$ flag for

each party i (including both clients and nodes). Additionally, it ensures that all the network packets intended to send for a given round is not send before or after that round to an honest protocol party. As a consequence, the environment can stop the entire protocol at anytime. As then no messages would be delivered anymore, stopping the entire execution does not leak any information to the environment.

Array Rounds := {**false**, . . . , **false**} // An array of length K + N + N
Round := 0
PartiesIncremented := 0
QUEUE = a queue where the incoming messages are stored

QueryRound() from $\mathcal{A}$ or $\mathcal{F}_{CRF}$ or party i:

return {Round, Rounds}

RequestRound() from party i:

return Rounds[i]

NextRound() from party i:

if Rounds[i] = **true** **then**
return "invalid action"
else
Rounds[i] ← **true**; PartiesIncremented += 1
if PartiesIncremented = K + N + N **then**
Round += 1; PartiesIncremented ← 0
Reset Rounds[j] := **false** $\forall j : 0 \leq j < n + m$
Forward all elements of QUEUE to $\mathcal{A}$; empty QUEUE

Upon receiving msg $(P, P_{next}, O, \text{round})$ **from Streams**

if round = Round **then**
ADD $(P, P_{next}, O, \text{Round})$ to QUEUE

Figure 5.2. Round Functionality $\mathcal{F}_{round}$

Randomness Beacon Functionality $\mathcal{F}_{CRF}$. We assume that each protocol party (including the adversary) has access to an incorruptible randomness beacon. In particular, future values of this randomness beacon are not known to the adversary. There are many public yet unpredictable randomness beacons are available in practice [63, 64]. To focus on the

building blocks that we provide, we abstract away from the cryptographic details of those constructions and assume such an ideal randomness beacon. We model this beacon with an ideal functionality $\mathcal{F}_{CRF}$ (see Fig. 5.3) that outputs each time a ℓ-long substring of an infinite random string beacon. Using that ℓ-length string a protocol party can derive the common path for the next ℓ rounds. Formally, we require the randomness beacon to be unpredictable before the protocol starts, as our adversary can only statically compromise parties; the beacon, however, can also be leveraged for resistance against slow dynamic compromisation as we discuss in Section 5.6.1.

crf = an infinitely long random string.

GetFunnels(ℓ):

$\mathtt{round}, \leftarrow \mathsf{QueryRound}()$
return $\{\mathsf{crf}[\mathtt{round}]\%\mathsf{K}, \ldots, \mathsf{crf}[\mathtt{round}+\ell-1]\%\mathsf{K}\}$

Figure 5.3. Randomness Beacon Functionality $\mathcal{F}_{CRF}$

Key registration functionality $\mathcal{F}_{RKR}$. The key registration functionality $\mathcal{F}_{RKR}$ is solely used by the subprotocol Π_{sub} from [60], which handles all cryptographic operations. Π_{sub} is treated in a black-box manner throughout this section. For completeness, we provide a description of Π_{sub} and $\mathcal{F}_{RKR}$ in Section 5.B.

Secure Channel Functionality $\mathcal{F}_{SCS}$. We also use the secure communications sessions functionality $\mathcal{F}_{SCS}$ from the work of Gajek et al. [76, Figure 4]. They show that $\mathcal{F}_{SCS}$ abstracts the TLS [77] protocol. It is crucial to note here that all the protocol parties in our model work in rounds, and therefore, $\mathcal{F}_{SCS}$ as well forwards all the messages to the $\mathcal{F}_{round}$ functionality instead of the environment; the $\mathcal{F}_{round}$ functionality in turn forwards those messages to the environment when the round ends.

5.2.3 The Core Protocol

First we present the core protocol that does not scale well with the number of users. Then in Section 5.2.4 we describe our complete protocol with horizontal scaling. Our protocol has

two kinds of parties — clients and nodes. So we define our protocol in two parts as well — clients and nodes. Additionally, the protocol parties as well as the adversary have access to the hybrid functionalities as described above (in Section 5.2.2).

Packet format. We use the Sphinx packet design [62,78] to ensure that all messages are end-to-end encrypted; we call them "onion packets". The Sphinx packet design also guarantees that an intermediate node, just by looking at a packet, does not learn any information beyond the routing information needed to forward the message to the next node — it hides the path length and the relay position of the node on the path, and the node does not learn anything other than the next nodes on the path.

Clients. Whenever a client wants to send a message m, she decides the delay d for every real message by picking a number from $[\frac{\ell}{2}, \ell - 1]$ following a distribution D. In general D can be any discrete probability distribution; however, in a typical setting we assume D to be a uniform distribution in $[\frac{\ell}{2}, \ell - 1]$. In Section 5.4 we provide other different instances of D for different settings.

The client derives the path of a packet based on the string returned by the randomness beacon. For a given round r if $\mathcal{F}_{CRF}$ returns the string $\{x_r, x_{r+1}, \ldots, x_{r+\ell}\}$, any onion packet constructed at round r will be constructed for the path of nodes $\{x_r, \ldots, x_{r+d}, R\}$ for a delay d and intended recipient R. The client will send the onion packet to node x_r at round r.

All other clients as well sends their packets to the node x_r at round r, since $\mathcal{F}_{CRF}$ returns the same node $\{x_r, x_{r+1}, \ldots, x_{r+\ell}\}$ to all the parties (clients, nodes, and the adversary) at round r.

Nodes. The nodes act similar to onion routers [49,54] except a node in our protocol accepts packets only in the rounds indicated by the randomness beacon. More formally, when a node receives a packet in round r, it checks if x_r matches its own id for a string $\{x_r, x_{r+1}, \ldots, x_{r+\ell}\}$ returned by $\mathcal{F}_{CRF}$ — if not, it rejects the packet. If x_r matches its id, the node with onion packets peels a layer of onion for each packet and forwards them to the next destination.

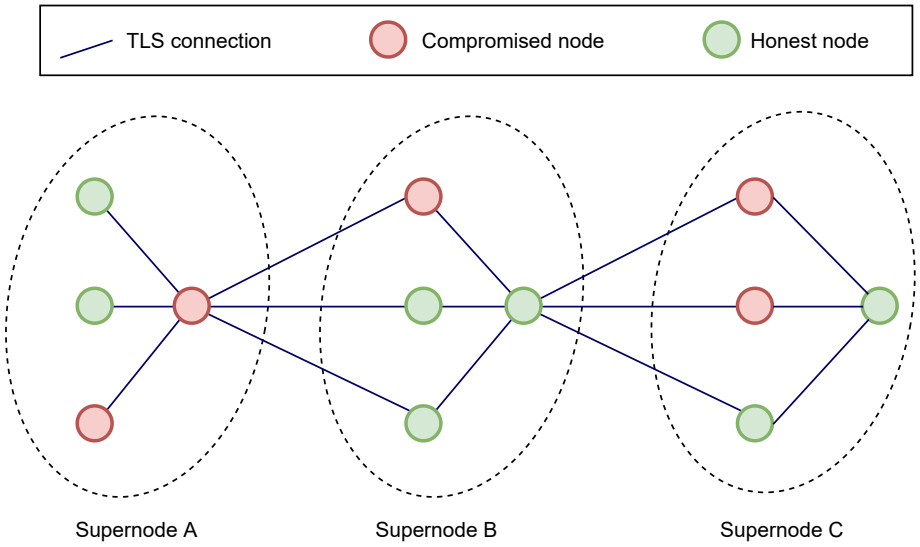


Figure 5.4. Supernode structures

5.2.4 Horizontal Scaling With Supernodes

One major bottleneck in the above protocol is the processing power of the nodes — the total number of clients the system can serve is restricted by the processing power of the weakest node. We introduce the concept of *supernode* where, instead of one node processing all the onion packets in a round, many nodes come together to share the processing load. A supernode consists of one *funnel node* that collects and mixes messages and many *compute nodes* that perform onion decryptions to prohibit linking. Figure 5.4 shows how a supernode is structured.

The supernode structures are not permanent, but conceptual: the same node can act as a funnel node or a compute node in different rounds. The funnel nodes are picked using the string $\{x_r, x_{r+1}, \ldots, x_{r+\ell}\}$ emitted by the randomness beacon $\mathcal{F}_{CRF}$. Each client picks the compute nodes uniformly at random (with replacement) from all available nodes for each hop of an onion packet independent of any other packet or any other hop of the same packet. All the packets in every odd round go to the designated funnel node. The funnel node immediately shuffles all the received packets and forwards them (without any cryptographic operation) to the compute nodes based on the next node information in the Sphinx packet

QUEUE = a FIFO queue.

SendMessage(msg, R**) from party** i

$r \leftarrow QueryRound()$
if round $\neq r$ **then**
  reject packet and exit
ADD (msg, R) to QUEUE

Upon new round from $\mathcal{E}$**:**

boolean flag := **RequestRound**() // defined in $\mathcal{F}_{round}$
if flag $\neq$ **true then**
  return "invalid action"
if round mod $2 = 0$ // compute round **then**
  while QUEUE is not empty **do**
    $(msg, R) \leftarrow$ dequeue QUEUE; $d \leftarrow DelayDistribution\left(\frac{\ell}{2}, \ell - 1\right)$
    $\{x_1, \ldots, x_d\} \xleftarrow{\$} I^d$; $p := \{x_1, \ldots, x_d, R\}$
    call `Process_new_onion`$(self, msg, d+1, p)$ from wrapper Π_T
$NextRound()$

Upon receiving a message msg **from** Π_{sub}**:**

Output "Message msg received" to $\mathcal{E}$

Figure 5.5. Client Protocol Design Π_{client}

header. Then the compute node removes one layer of the onion packet, and forwards the packet immediately to the next designated funnel node.

Funnel nodes act as mix nodes for the messages. All messages will meet in the same funnel nodes as their paths are coordinated by the randomness beacon. Specifically, a node only acts as a funnel node if the randomness beacon determines that it is the funnel node for the current round.

Compute nodes act similar to onion routers [49, 54] — in every even round a node with onion packets peels a layer of onion for each packet and forwards them to the next designated funnel node.

The packets are onion encrypted only for the compute nodes, not for the funnel nodes. Additionally, we assume authenticated and encrypted channel between each pair of nodes which is realized by the $\mathcal{F}_{SCS}$ functionality. We assume that all the nodes are optimized

Π_T:`Process_new_onion`$(self, msg, d+1, p)$

Call `Process_new_onion`$(self, msg, d+1, p)$ in the subprotocol Π_{sub}.
Intercept the network packet *packet* and send it to Π_{rer}.

Π_T: `Forward_Onion`(O)

Call `Forward_Onion`(O) in the subprotocol Π_{sub}.
Intercept the network packet *packet* and send it to Π_{rer}.

Π_{rer}: **Upon a packet** *packet*

$\_, funnel \leftarrow$ GetFunnels(2) // Select next funnel node
Send *packet* over $\mathcal{F}_{SCS}$ to *funnel*.

Figure 5.6. $\Pi_T$and Π_{rer}

for communication, since we can increase the number of compute nodes if we need more processing.

The protocol run by each honest client is defined in Fig. 5.6; and the protocol run by each honest node is defined in Fig. 5.7. In Section 5.6.2 we discuss the challenges when the clocks are not properly synchronized, and present how to handle such loose synchronization.

It is important to observe that the following scenario is equivalent to two messages going through an honest mixnode in the core protocol: two messages are processed by some honest nodes (not necessarily same) in round r, and then both of them goes through the same honest funnel node in round $r+1$. In the case mentioned above, the two messages achieve "mixing" even if the whole network before and after that is compromised. In Figure 5.8, we pictorially show the possible cases when two messages can mix (or not).

5.3 Security Proof

5.3.1 Universal Composability

For proving security, we first prove an intermediary representation of Streams that does not rely on cryptographic operations but on shared memory. This abstraction $\mathcal{F}_{\text{Streams}}$ (called an *ideal functionality*) is carefully crafted such that all attacks on Streams can be mounted

INPUT_QUEUE = a queue where the node stores incoming messages.
OUTPUT_QUEUE = a queue where the node stores outgoing messages.
nodeID := a unique ID in $[0, \mathsf{K}-1]$

Upon input message (onion packet O):

ADD O to INPUT_QUEUE

Upon new round from $\mathcal{E}$:

boolean flag := **RequestRound**() // defined in $\mathcal{F}_{round}$
if flag $\neq$ **true then**
  return "invalid action"
$funnel$:= GetFunnels(1)
if round mod 2 = 1 AND nodeID = $funnel$ **then**
  Π_{funnel}
else if round mod 2 = 0 **then**
  Π_{worker}
swap INPUT_QUEUE and OUTPUT_QUEUE.
NextRound()

Π_{funnel}**:**

Shuffle OUTPUT_QUEUE
while OUTPUT_QUEUE is not empty **do**
  $O \leftarrow$ dequeue the first element from OUTPUT_QUEUE
  Forward O to $\mathcal{F}_{SCS}$

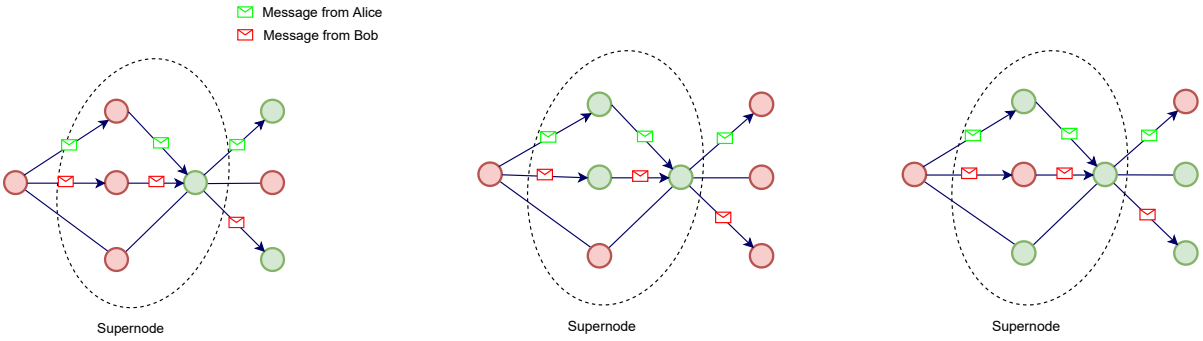
Π_{worker}**:**

while OUTPUT_QUEUE is not empty **do**
  $O \leftarrow$ dequeue the first element from OUTPUT_QUEUE
  call $Forward\_Onion(O)$ from the subprotocol wrapper Π_T

Figure 5.7. Node Protocol Design

on $\mathcal{F}_{\mathsf{Streams}}$ as well. More generally, we say that the ideal functionality $\mathcal{F}$ *is realized by* a protocol Π if all attacks (within the execution model) that can be mounted on Π can be translated to attacks on Π. An ideal functionalities, like $\mathcal{F}_{\mathsf{Streams}}$, can abstract away from cryptographic details while faithfully modeling all weaknesses of a protocol. In turn, the absence of attack vectors in the ideal functionality implies the absence of attack vectors in the protocol that realizes that ideal functionality. Hence, an ideal functionality can be an accurate characterization of a protocol's security properties.

We use the *universal composability framework* [75] to formulate our ideal functionality, as it enables us to re-use a cryptographic analysis by Kuhn et al. [60] of a variant of the Sphinx



(a) Message from Alice and Bob do not mix with each other at the supernode because the malicious compute nodes know the recipient of a message.

(b) Messages from Alice and Bob mix with each other when they pass through honest compute nodes and then an honest funnel in the supernode.

(c) Messages from Alice and Bob do not mix with each other if both of them do not pass through honest compute nodes in the same round.

Figure 5.8. Cases when two messages mix (or not) in supernode structures

protocol. Hence, we do not need to argue about details of the underlying cryptographic algorithms.

In order to achieve a composable notion of realization, the UC-framework additionally requires black-box simulatability in the following sense: a protocol Π *UC-realizes* an ideal functionality $\mathcal{F}$ iff there is a black-box simulator $\mathtt{S}$ such that any interactive distinguisher (called the *environment*) that has access to the API of the protocol and that controls (or eavesdrops on) the network cannot distinguishing the interaction with Π from the interaction with $\mathcal{F}$ and the black-box simulator $\mathtt{S}$.[3] To achieve universal composability, the simulator $\mathtt{S}$ is not allowed to alter the API called but only to simulating network messages.

The composability result of the UC framework makes a statement about ideal functionalities $\mathcal{F}$ that are UC-realized by protocols Π. The composability states that any parent Π protocol, that uses Π over its API as a subprotocol UC realizes the abstracted protocol where Π uses $\mathcal{F}$ over its API.

[3]↑For the experts, we are simplifying the presentation here, as the completeness of the dummy attacker implies that the network attacker is irrelevant.

5.3.2 An Ideal Functionality for AC Protocols

The ideal functionality $\mathcal{F}_{\mathsf{Streams}}$ basically acts as as trusted third party to whom users tell that they would like to anonymously send a message. This trusted third party leaks as much information as Streams would leak. Due to the regular meeting points at funnel nodes and TLS protection, $\mathcal{F}_{\mathsf{Streams}}$ does not need to leak which onion is sent from which compute node to which compute node, as long as the party that sends the onion and the next subsequent funnel node are honest (see Figure 5.4).

Removing this very leakage of which onion is sent to whom enables us to prove a strong shuffling property for Theorem 5.3.1, pairwise unlinkability with overwhelming probability (see Definition 5.1.1).

inputBuffer[] an array of queues to store messages for nodes
crf = an infinitely long random string
dlvrCM, *dlvrHM*, *queue*, *DB* are hashmaps.
round := 0, *newRound*[] := {false, false, ...}, *partyCount* := 0

Upon new round from $\mathcal{E}$ for party P:

if *newRound*(P) = **true** **then** return "invalid action"
set *newRound*(P) := **true** ; *partyCount*+ = 1
if round is odd (funnel round) AND P is a client **then**
  $(m, R, t) \leftarrow$ dequeue *inputBuffer*[P]
  $d \leftarrow$ *DelayDistribution*$(\frac{\ell}{2}, \ell - 1)$; $\{x_1, \ldots, x_d\} \xleftarrow{\$} I^d$
  if $!\exists x \in \{x_1, \ldots, x_d\}$ such that $x_a \in \mathtt{I_h}$ **then**
    Send $(m, x_1, \ldots, x_d)$ to $\mathcal{S}$
  else
    let $x_a :=$ the **first** honest party on the path $\{P, x_1, \ldots, x_d\}$
    Send $(q, x_1, \ldots, x_a)$ to $\mathcal{S}$ where $q \xleftarrow{\$} \mathcal{M}$
    store $(q, x_a, m, x_{a+1}, \ldots, x_d, R)$ in *queue*(round + a)
if round is even (compute round) AND *partyCount* = N + K **then**
  SendInformation()
NextRound(P)

Upon input message (m, R, t) from $\mathcal{E}$ for party P:

inputBuffer(P)+ = 1
if round $\neq t$ **then** reject packet and exit
Add (m, R, t) in *inputBuffer*[P]

Upon receiving a message m for party P:

Output "Message m received" to $\mathcal{E}$

Figure 5.9. Ideal functionality $\mathcal{F}_{\mathsf{Streams}}$

```
SendInformation()
  y := crf[round + 1]%K
  for each (q, x_a, m, x_{a+1}, ..., x_d, R) ∈ queue(round) do
    Remove (q, x_a, m, x_{a+1}, ..., x_d, R) from queue(round); link := q
    if y ∈ I_h then link := ⊥
    let x_γ be the next honest node on the path {x_{a+1}, ..., x_d}
    if the is no such x_γ then
      Add (link, m, x_{a+1}, ..., x_d, R) in a temporary queue Q
    else
      Add (link, q, x_{a+1}, ..., x_γ) in Q for q ←$ M
      Add (q, x_γ, m, x_{γ+1}, ..., x_d, R) to queue(round + γ − a)
  Shuffle the elements of Q and send them to S
```

Figure 5.10. Leakage From Ideal functionality $\mathcal{F}_{\mathsf{Streams}}$

Formally, the ideal functionality $\mathcal{F}_{\mathsf{Streams}}$ provides API calls for when clients want to send a message and they react to network messages. Moreover, as we consider a round-based protocol and the UC-framework is a sequential activation framework (to simplify the analysis), we formally need a "new round" API call.

The ideal functionality expects input messages of the form (msg, R, t). As the protocol works in rounds, $\mathcal{F}_{\mathsf{Streams}}$ stores the input messages in an input queue. Upon the "new round"-command, an element from the input queue is processed. When processing an input, the ideal functionality $\mathcal{F}_{\mathsf{Streams}}$ checks which message only has compromised parties $x_i \notin \mathtt{I_h}$ on its path. For those cases, the ideal functionality leaks the message to the simulator $\mathcal{S}$. Otherwise, $\mathcal{F}_{\mathsf{Streams}}$ provides a temporary identifier (in the form of a random integer) to $\mathcal{S}$ in place of a message, along with the segment of the path until the next honest compute node. When the round corresponding to that compute node comes, $\mathcal{F}_{\mathsf{Streams}}$ again provides a new temporary identifier along with the next segment of path. When $\mathcal{F}_{\mathsf{Streams}}$ switches the temporary identifiers, if there are not honest funnel before or after the honest compute node, it provides the mapping between the old and the new identifiers to allow $\mathcal{S}$ to link between packets. Here we slightly over-approximate the leakage by not distinguishing between honest and compromised recipients, because in some protocol setting (anonymous broadcast) or anonymity notion (sender anonymity) the adversary can anyway see the message in plaintext once it comes out of the protocol. We formally present the ideal functionality in Fig. 5.9.

Even though we all $\mathcal{F}_{\mathsf{Streams}}$ an ideal functionality, we still keep the round functionality $\mathcal{F}_{round}$ as a hybrid functionality to reflect the desired properties of AC protocols.

5.3.3 Ideal Functionality $\mathcal{F}_{\mathsf{Streams}}$ as Trusted Third Party Stop-and-go Mix

Next we show that, at the expense of latency, our ideal functionality behaves as a trusted third party (TTP) anonymizer. If two messages stay together in $\mathcal{F}_{\mathsf{Streams}}$ for a sufficiently long time (poly-logarithmic in the security parameter), they get shuffled — as if they stayed in a TTP for several rounds and then get delivered to the recipient. However, if the delay distribution for the messages is predictable, the adversary can still break anonymity, and even a TTP anonymizer cannot defend against that. Later in this section, we provide analysis about how to pick the delay distribution to achieve strong sender or relationship anonymity.

Formally, Theorem 5.4.1 shows that this ideal functionality $\mathcal{F}_{\mathsf{Streams}}$ is UC-realized by Streams. As a reminder, $\mathcal{F}_{\mathsf{Streams}}$ captures all attacks and all leakage of the protocol Streams while abstracting away from all cryptographic operations and directly leaks the information that the protocol leaks. $\mathcal{F}_{\mathsf{Streams}}$ shows that Streams essentially solely leaks information in three cases by characterizing the leakage in $\mathcal{F}_{\mathsf{Streams}}$: $\mathcal{F}_{\mathsf{Streams}}$ leaks when parties send their messages (be it dummy or real messages); $\mathcal{F}_{\mathsf{Streams}}$ leaks the message if a message is sent to a compromised recipient; $\mathcal{F}_{\mathsf{Streams}}$ leaks that a message is sent from an inner node (a router) to a client.

Theorem 5.3.1 (Pairwise unlinkability of $\mathcal{F}_{\mathsf{Streams}}$)**.** *If the amount of compromised nodes is a constant fraction* $\frac{\mathsf{c}}{\mathsf{K}} < 1$*, $\mathcal{F}_{Streams}$ provides pairwise unlinkability of messages over $\mathfrak{L}$ rounds up to probability δ as in Definition 5.1.1, where $\delta < \gamma^{\mathfrak{L}}$ with $\gamma = 1 - \left(\frac{K-c}{K}\right)^3$.*

For conciseness of presentation, we postpone the proofs in Section 5.A. The key idea is that two messages get shuffled if they go through an honest funnel node right after going through honest compute nodes. It is not necessary that they pass through the same honest compute node, however, they need to pass through some honest compute nodes in the same round. If $\mathfrak{L}$ is the above theorem is polylogarithmic δ becomes negligible, which gives us the following corollary.

Corollary 1. *Given a constant fraction* $\frac{\mathsf{c}}{\mathsf{K}}$*, in the presence of any adversary* $\mathcal{S}$*, if two arbitrary messages stay together in the protocol* $\mathcal{F}_{\mathsf{Streams}}$ *for* $\mathfrak{L} \in \omega(\log \eta)$ *rounds they are shuffled with an overwhelming probability.*

5.4 Security Analysis

In this section we formally analyze the security of our protocol Streams against a global passive adversary that can passively compromising (the compromised parties still follow the protocol) some portion of the nodes. We discuss the required integrity measures for our protocol against active adversaries in Section 5.6.1.

5.4.1 Abstraction Proof for Streams

Recall that formally Streams runs in the $\mathcal{F}_{CRF}, \mathcal{F}_{RKR}, \mathcal{F}_{SCS}, \mathcal{F}_{round}$ hybrid model. Our ideal functionality $\mathcal{F}_{\mathsf{Streams}}$ absorbs the hybrid functionalities $\mathcal{F}_{CRF}$, $\mathcal{F}_{RKR}$ and $\mathcal{F}_{SCS}$ completely. However, we keep the $\mathcal{F}_{round}$ functionality untouched.

Kuhn et al. [60] show that under standard cryptographic assumptions there is a protocol Π_{sub} in the $\mathcal{F}_{RKR}$-hybrid model that UC realizes $\mathcal{F}_{sub}$.

Theorem 5.4.1. *For any subprotocol* Π_{sub} *in the* $\mathcal{F}_{RKR}$*-hybrid model that UC realizes* $\mathcal{F}_{sub}$*, the anonymity protocol* Streams *from Section 5.2 using the subprotocol* Π_{sub} *in the* $\mathcal{F}_{CRF}$*,* $\mathcal{F}_{RKR}$*,* $\mathcal{F}_{SCS}$*,* $\mathcal{F}_{round}$*-hybrid model UC-realizes* $\mathcal{F}_{\mathsf{Streams}}$ *in the* $\mathcal{F}_{round}$*-hybrid model.*

The key idea is to utilize (in a black-box reduction) the UC-realization proof of Π_{sub} such that in the proof the subprotocol's ideal functionality $\mathcal{F}_{sub}$ can be considered. This ideal functionality $\mathcal{F}_{sub}$ is used to abstract away from any cryptographic operations. The second key insight is that the attacker (and the simulator) can perfectly predict how many onions are in the protocol and when each party sends a message. So, only if the recipient is compromised or a message is sent to a client (or the input buffer is full) information is leaked from the protocol. In those cases, the ideal functionality $\mathcal{F}_{\mathsf{Streams}}$ indeed leaks information such that the simulator can faithfully (and indistinguishably) simulate the network traffic. The full proof is postponed to Section 5.A.1.

5.4.2 Pairwise Unlinkability of Streams

Since $\mathcal{F}_{\mathsf{Streams}}$ provides pairwise unlinkability of messages for $\delta < \beta^{\mathfrak{L}}$ over $\mathfrak{L}$ rounds, as a corollary to Theorem 5.4.1 and Theorem 5.3.1 we can state the following security theorem for our protocol Streams.

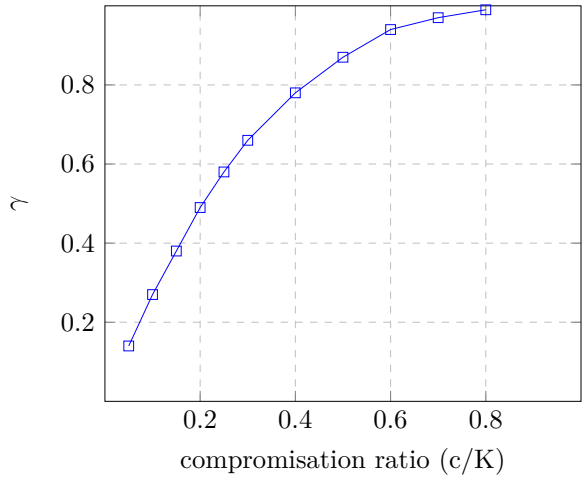
Theorem 5.4.2 (Security of Streams)**.** *For any subprotocol Π_{sub} in the $\mathcal{F}_{RKR}$-hybrid model that UC realizes $\mathcal{F}_{sub}$, given a constant fraction $\frac{\mathsf{c}}{\mathsf{K}} < 1$, Streams (using Π_{sub}) provides pairwise unlinkability of messages over $\mathfrak{L}$ rounds up to probability δ as in Definition 5.1.1, where $\delta < \gamma^{\mathfrak{L}}$ where $\gamma = 1 - \left(\frac{\mathsf{K}-\mathsf{c}}{\mathsf{K}}\right)^3$.*

For any $\mathfrak{L} \in \omega(\log \eta)$ with a security parameter η, δ is negligible, and all pair of messages that stays together in the protocol for at least $\mathfrak{L}$ rounds, get shuffled with overwhelming probability.

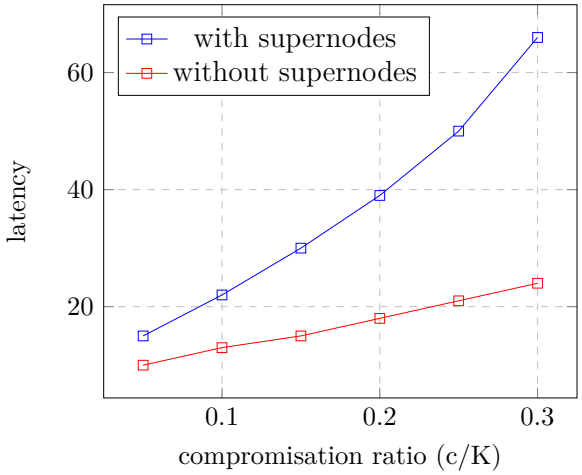
Note that, in our formal description of Streams, we have separate funnel rounds and compute node rounds and users can send messages only in the compute node rounds. However, the compute nodes do not really need to wait for the whole round to forward the packets, they can forward those packets as and when they process them. Splitting funnel round and compute round is completely UC-artifact to conform with the communication model there. Therefore, in real world we can merge funnel round and compute node round into a single round. Suppose, the overall round duration is T milliseconds. Every party (clients and other nodes) will forward all the packets to that chosen funnel node in that round. The funnel node will shuffle all the packets received in that round, and forward them to compute nodes immediately after the round is over. During the next round, the compute nodes processes and forwards the processed packets to the next funnel node. Henceforth, by "delay of ℓ rounds" we mean the overall delay of ℓ *merged*-rounds which is actually 2ℓ rounds in UC communication model.

We show that $\gamma = 1 - \left(\frac{\mathsf{K}-\mathsf{c}}{\mathsf{K}}\right)^3$ (refer to Section 5.A), γ is conceptually the proportion of supernodes where the two messages cannot mix. In Figure 5.11a we plot the relationship between γ and $\frac{\mathsf{c}}{\mathsf{K}}$.

If we want to have the same level of concrete security as without supernode, we need to increase ℓ, or with similar ℓ the protocol can only be resilient against lesser fraction of



(a) The fraction of compromised nodes $\frac{c}{K}$ (x-axis) vs. the probability γ (y-axis) of two messages not mixing when they pass through a randomly chosen supernode.



(b) The fraction of compromised nodes $\frac{c}{K}$ (x-axis) vs. the required latency ℓ (y-axis) to achieve one in a trillion security ($\delta < 2^{-40}$).

Figure 5.11. Effective security of supernodes

compromised nodes. However, one advantage of this construction is that the cost or overhead does not increase linearly with or depend on the number of clients. In Fig. 5.11b, we compare the latency overhead needed for for our protocol (with supernodes) to achieve one in a trillion security with that of our core protocol without supernodes (refer to Theorem 5.4.3). For $\frac{c}{K} \leq 0.2$ the latency only doubles with supernodes to achieve the same level of security. Even though, supernodes provide scalability at the cost of security, the δ value still decreases exponentially with latency. We show the relationship between δ and ℓ in Fig. 5.12.

5.4.3 Pairwise Unlinkability for the Core Protocol

We also want to consider the scenario where we do not need to scale horizontally (e.g., the nodes are as powerful as network routers or the total number of users is less than few thousands) — in that case we do not need the supernode construction and the core protocol described in Section 5.2.3 is sufficient. The following result gives us an important insight about how much security is degraded to achieve scalability.

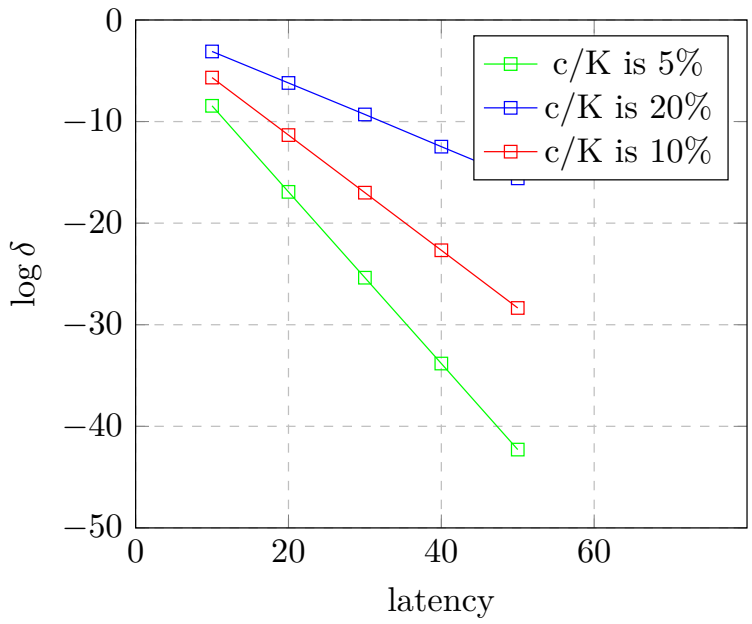


Figure 5.12. latency vs. $\log \delta$ for different values of $\frac{c}{K}$. The linear decrease in $\log \delta$ means exponential decrease in δ value.

Theorem 5.4.3 (Pairwise unlinkability of the Core protocol)**.** *For any subprotocol* Π_{sub} *in the* $\mathcal{F}_{RKR}$*-hybrid model that UC realizes* $\mathcal{F}_{sub}$*, given a constant fraction* $\frac{c}{K} < 1$*, the core protocol (using* Π_{sub}*) described in Section 5.2.3 provides pairwise unlinkability of messages over* $\mathfrak{L}$ *rounds up to probability* δ *as in Definition 5.1.1, where* $\delta < \left(\frac{c}{K}\right)^{\mathfrak{L}}$.

Similar to other proofs, we postpone this proof to Section 5.A.2.

5.5 Performance Evaluation

In this section we provide an instantiation of Streams with using a prototype implementation and evaluate the performance. We first describe the implementation details and system considerations, then we present our experimental results.

5.5.1 Prototype Implementation And System Considerations

To evaluate Streams we have developed a proof-of-concept implementation in approximately 5000 lines of Go code (v1.15). Our implementation[4] builds upon the existing Loopix implementation [79] for the crypto and sphinx packet implementations; then we add our own implementation for funnel and compute nodes.

Synchronization. For the prototype implementation we consider a global clock that every protocol party (clients and nodes) follows. However for real deployments, the global clock can be replaced with local clocks in combination with the idea of loose synchronization technique described in Section 5.6.2. In Section 5.6, we also discuss about how to make use of the randomness beacon as the synchronizer.

About Rounds. We decide the round duration based on the load on the system, or more specifically, how many onion packets are there in the system at any given point of time. We are going to demonstrate later in this section that the amount of time it takes to process onion packets is proportional to the number of packets in our system.

Random Shuffle. We implement the Fisher-Yates shuffle [80] to achieve in-memory shuffle of n elements with $\Theta(n)$ computational complexity. This algorithm requires a continual source of randomness, and for that purpose each funnel node can use a locally stored random number table.

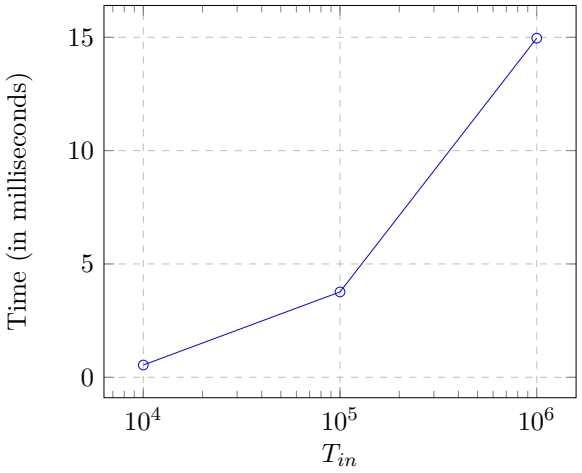
5.5.2 Processing Capacity of Funnel Nodes

We first evaluate how easily a funnel node can process different numbers (10K, 20K, 50K, 100K etc) of onion packets — because that (plus the communication latency between funnel and compute nodes) will dictate the round duration if we want the protocol to handle those many packets in every round. To evaluate the processing capacity of a funnel node, we run a standalone funnel node and send varying number of onion packets to that funnel node. On the funnel node we measure how much time is takes by the TLS layer to process all of those packets, as well as how much time is taken to run the the shuffle algorithm.

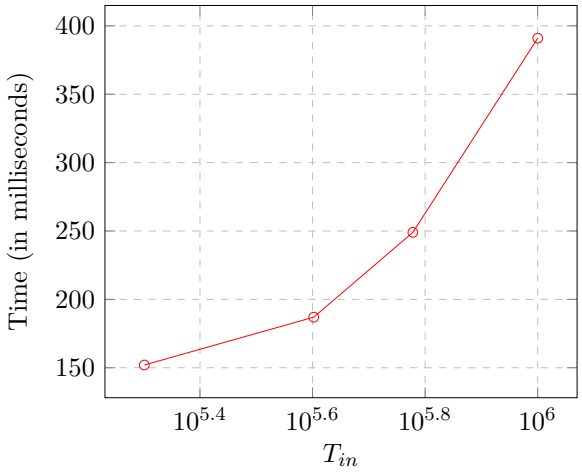
[4]↑The implementation is available at the following anonymized Google drive link https://drive.google.com/drive/folders/10EjJGqo0ZzHrLm5V78dSRDdOhR1FQHd0

To measure our benchmarks, we run the funnel node pragram on a machine which has 48 Intel Xeon Silver 4116 processors (3 GHz) with 128 GB RAM. We plot our findings in Figure 5.13. All the measurements are average of 10 runs approximated to the nearest integer. Since we achieve in-memory shuffle using Fisher-Yates algorithm, the observed shuffle time remains less than 15 milliseconds even for 1 million packets, even though the whole shuffle protocol needs to run in a single thread.

On the other hand, processing 150K onion packets over TLS takes more than 150 milliseconds. The overhead involves AES encryption/decryption for TLS, and handling multiple TLS threads to ensure one thread does not overwrite a packet from another thread. For processing packets received via TLS, we allow the server to spawn up to 20 threads. This experiment shows that the dominant factor in deciding the round duration is TLS processing. This overhead can be further improved by having a more optimized implementation to handle the TLS threads.



(a) Number of packets (T_{in}) sent to a funnel node vs. the amount of times in microseconds taken to run Fisher Yates shuffle.



(b) number of packets (T_{in}) sent to a funnel node via TLS (x-axis) vs. the amount of times taken to process those packets (y-axis).

Figure 5.13. Onion packets processing at funnel nodes

5.5.3 Processing Capacity of Compute Nodes

Now we want to evaluate how many onion packets can be processed by a single compute node in a given amount of time. Given a round duration (which is chosen based on the total number of messages in the system), this measurement will give us an estimate of how many compute nodes we need in the whole system to process all the onion packets. To that end, we run a standalone compute node, then give the node different number of onion packets to process, and measure the time spent to process those packets. For this experiment, we run our compute node on a system with the same configuration as the system we used for the experiment with funnel nodes (48 Intel Xeon Silver 4116 processors with 128 GB RAM).

In Figure 5.14 we plot a graph between number of onion packets given to a compute node vs. time taken in milliseconds to process those packets (y-axis). We take the measurements by repeating the experiment 10 times and taking an average of those.

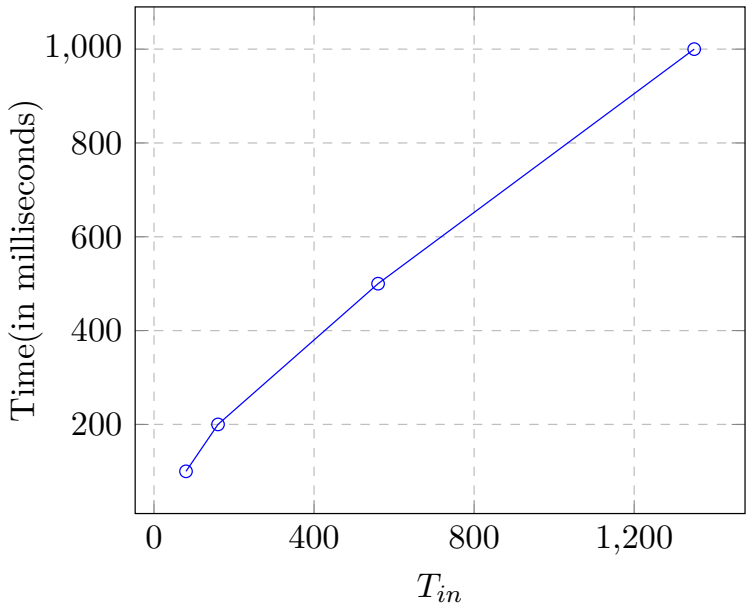


Figure 5.14. number of packets (T_{in}) sent to a node per round (x-axis) vs. time taken in milliseconds for the worker node to process those packets (y-axis)

If there are on average 500K packets in the system at any point in time, the funnel node takes $<$ 250 milliseconds to process them (refer to Fig. 5.13), so we can pick round size as 400 milliseconds (adding 80 milliseconds for two communication hops, one funnel to

compute node, and second hop compute node to funnel). In that case, each compute node has around 300 milliseconds for processing the onion packets — therefore, we need around 1250 compute nodes in the system (each compute node processing around 400 packets, refer to Figure 5.14).

5.5.4 End-to-end Latency Evaluation

We evaluate the end-to-end latency offered by our protocol in the following way: we determine the round duration based on the total number of messages we want our system to support; and from Fig. 5.11b we know the number of rounds required to achieve $\delta < 2^{-40}$. If we consider that almost 20% of the total nodes are compromised, we need $\ell = 40$. For different number of messages, in Table 5.1 we summarize the round length (in milliseconds) required for our protocol and the end-to-end latency offered.

We want our system to handle 1 million messages at any point in time. Therefore, we pick a round duration of 500 milliseconds; with $\ell = 40$ to achieve $\delta < 2^{-40}$, the estimated end-to-end latency is about 20 seconds. If the system can process 1 million messages per round, it can allow about 25K new messages on an average in every round (with $\ell = 40$). Which means our system can support 25K users even if every user sends a message in each round. We want to emphasize that an optimized implementation deployed on servers with high communication capability will improve the numbers significantly.

Table 5.1. End-to-end latency offered by Streams for different load on the system. The leftmost column represents the total number of messages (T) at any given point of time, the second column round length (R) in milliseconds, the third column (L) the end-to-end latency in seconds (assuming $\frac{c}{K} = 0.2$ or $\ell = 40$), and the fourth column the number of users (N) that can be supported by the protocol assuming each user sends a message in every round. We assume a network latency of around 40 milliseconds for each hop.

Load(T)	round size (R)	latency (L)	N
200K	250 ms	10 sec.	5K
600K	350 ms	14 sec.	15K
1M	500 ms	20 sec.	25K

5.6 Discussion

5.6.1 Against Active and Adaptive Attackers

Here we write the integrity measures that we employ to protect the protocol against active attacks. We incorporate the exact same techniques as Loopix [1] for the protocol's overall integrity.

Packet Format. Since our protocol makes use of Sphinx [62] packet format that comes with confidentiality (including padding), as well as message integrity.

Kill Alice Attack. Similar to Loopix, in our protocol users can detect such attacks by sending messages to themselves (called "loop messages"). In case, the adversary decides to drop all messages from a specific user Alice, Alice will not receive the loop messages and she will know that she is under attack.

DoS Attacks. We do not consider DoS attacks unless anonymity can be broken using DoS attacks. Although our security proof is mainly for passive adversary, the anonymity game allows the adversary to stop the protocol at any point in time – which allows the security proof to consider DoS attacks that can break anonymity.

Slow Dynamic Compromisation. Our core protocol uses a randomness beacon to make the future route beyond the ℓ-th successor unpredictable. Hence, attacker that can dynamically compromise parties but takes more than ℓ rounds for each compromisation will not be able to get in a specific message's path with high certainty by compromising only one node. However, this problem is not present in the version of the protocol with supernodes.

5.6.2 Resiliency Improvement For Loose Synchronization

In our protocol description in Section 5.2 we assumed that all the protocol parties are perfectly synchronized. However, it is difficult to achieve such synchronization in practice. Here we discuss how to relax that assumption by allowing each protocol party to follow their own local clock.

We assume that the maximum difference between two local clocks of the nodes is bounded by μ milliseconds. The clients do not need to keep track of rounds at all, and can send

messages to the system whenever they want. A client sends an onion packet to the first compute node on the onion path. The compute node based on its local clock can decide which funnel node to forward the packets to. As long as μ is lower than a few hundred milliseconds, we can add μ in the computation of round duration to handle the synchronization gap among the nodes. We can still have a reference global clock which the nodes can synchronize their local clocks with from time-to-time. We only need equivocation protection from that global clock, the protocol does not depend on that clock for anonymity.

However, suppose nodes (at most 10% for example) in the system have a difference of more than few hundred milliseconds with the reference global clock. Such unsynchronized compute nodes can send packets to wrong funnel nodes. If a node receives an onion packet that it is not supposed to receive (probably a dishonest or badly synchronized compute node has sent the packet to a wrong funnel node), the node just forwards the packet to the correct compute node (according to the onion packet header) at the end of the round. Therefore, the protocol still functions properly, although the latency needs to be increased based on the amount of such unsynchronized nodes (and compromised nodes) to maintain the same level of anonymity.

With the above modified approach, A node does not have to derive at which round it should act as a funnel node or compute node. For all the onion packets (according to the onion headers) if it is the intended compute node, it acts as a compute node; for all the rest of the packets it acts as a funnel node and forwards them to the next corresponding compute nodes at the end of the round.

5.6.3 Pairwise Unlinkability and Anonymity

We have shown that Streams provides pairwise unlinkability of messages as in Definition 5.1.1. In this section we compare the notion with similar notions in the literature and relate it to anonymity notions, such as sender anonymity and relationship anonymity.

Our notion of pairwise unlinkability is conceptually closely related to *tail indistinguishability* by Kuhn et al. [60]. The main difference is that in their definition packets are required to meet in a node that processes them cryptographically. Since our supernode structure

splits "nodes" into compute nodes and funnel nodes, our notion of pairwise unlinkability is not tied to packets that are assumed to meet, but covers all packets. In this sense it follows the older notion of unlinkability of Kate et al. [61], but extends it with explicit times messages enter and leave the system.

Sender anonymity. One common anonymity notion, *sender anonymity*, states that the recipient of a message cannot distinguish whether the message originated in one sender over another sender, even for a pair of potential senders of the adversary's choice. This notion closely resembles pairwise unlinkability with one key difference: sender anonymity typically talks about a single challenge message, not about a pair of messages; this can be overcome by requiring a degree of bandwidth overhead, such as ensuring all senders communicate regularly and can send dummy messages to confuse the adversary. However, even requiring dummy messages to be sent, an adversary might still deduce the challenge sender from timings alone.

If, say, the adversary observes Alice sending a message in round t and Bob sending a message in round $t+2$, the arrival time of the challenge message together with the distribution of the latency might tell the adversary who of them is more likely to have sent the challenge message. In the simplest example, for a constant latency, the adversary could immediately exclude one of them from being the challenge sender.

Note that this apparent attack is independent of how the protocol in question achieves anonymity; it even applies if the messages are kept in a trusted third party for the same amount of time.

Relationship anonymity. A similar notion states that if two senders send one message each to two receivers, a third party is unable to decide which sender talks to which receiver significantly better than purely guessing. Loopix calls this property *Sender-Receiver Third-party Unlinkability*. Given that the two messages in question are sent in the same round and that both senders choose a sufficiently large latency from the same distribution, pairwise unlinkability immediately implies this anonymity property.

If the challenge senders send their messages in different rounds we achieve a weaker, quantitative form of this property (akin to differential privacy), where the degree of anonymity depends directly on the difference in rounds and the latency distribution.

5.6.4 Anonymity vs. Latency Trade-off in Parallel Mixnets

One common bottleneck in traditional mixnet based systems is the processing power of the nodes — the total number of users the system can serve is restricted by the processing power of the weakest node. To avoid this issue, many systems [1, 2, 31, 33, 56] employ some form of parallel mixnets. However that approach does not always provide provable security. Actually, it is theoretically demonstrated [81] in literature that achieving provable guarantees with such a strategy can be really expensive in terms of latency and bandwidth overhead. In practice, protocols such as Yodel [56], Karaoke [31], Stadium [33], Vuvuzela [2] only achieves anonymity in the differential privacy sense at the expense of latency overhead. Atom [32] can achieve provably strong anonymity while scaling for millions of users with a really high end-to-end latency ($\tilde{2}8$ minutes). Typically the latency needs to grow for those protocols with number of users to maintain the same level of anonymity.

With supernode construction in Streams the latency increases more gracefully with the number of users — up to a few millions users, the latency does not need to increase even though the anonymity level remains the same. That demonstrates that our protocol with supernodes is much better suited to scale with number of users. Other protocol like Loopix [1], Tor [49], Hornet [14] can employ this supernode structures with minor modifications, independent of their original routing strategy, to achieve better anonymity guarantees.

5.6.5 Application Scenarios

Overall, our performance analysis clearly demonstrates that Streams can scale well with a large number of users. Beyond the traditional mixnet applications such as anonymous e-mailing, we find it useful to applications such as network-level anonymity for publishing blockchain transaction [82]: as the consensus process already takes up to a few minutes in these environments, a latency delay of a few seconds for provable network anonymity can be acceptable.

5.A Postponed proofs

5.A.1 Streams UC-realizes Ideal Functionality $\mathcal{F}_{\mathsf{Streams}}$

We stress that it makes our result stronger that we cast our ideal functionality in the $\mathcal{F}_{round}$-hybrid model. It is straightforward to let $\mathcal{F}_{\mathsf{Streams}}$ additionally absorb $\mathcal{F}_{round}$.

Theorem 5.4.1. *For any subprotocol Π_{sub} in the $\mathcal{F}_{RKR}$-hybrid model that UC realizes $\mathcal{F}_{sub}$, the anonymity protocol Streams from Section 5.2 using the subprotocol Π_{sub} in the $\mathcal{F}_{CRF}$, $\mathcal{F}_{RKR}$, $\mathcal{F}_{SCS}$, $\mathcal{F}_{round}$-hybrid model UC-realizes $\mathcal{F}_{\mathsf{Streams}}$ in the $\mathcal{F}_{round}$-hybrid model.*

Proof. We show the theorem via a series of game hops, starting with the protocol Pi and an arbitrary network adversary $\mathcal{A}$. With delta changes in each game, in the final game we end up with the ideal functionality $\mathcal{F}_{\mathsf{Streams}}$ and a simulator $\mathcal{S}$. As $\mathcal{F}_{CRF}$ does not send messages to the environment and is not accessible to the environment, it can be easily absorbed by the ideal functionalities. For brevity, we hence neglect it in the subsequent argumentation.

Game 1. *In this game, we consider the original protocol Streams execution with the network attacker $\mathcal{A}$ and the environment $\mathcal{E}$. The protocol follows the code in Figure 5.6 and Fig. 5.7.*

Now, we design a game and a protocol where the subprotocols associated with onion processing are replaced with ideal functionality from [60].

Game 2. *Instead of calling the protocol subroutines from Π_{sub}, our protocol Streams now calls the ideal functionality $\mathcal{F}_{sub}$ from Kuhn et al. [60] (for completeness also in the appendix Figure 5.16). Moreover, the attacker is replaced by a variant of the simulator S_{sub} from Kuhn et al.*

- ***The simulator*** *S^*_{sub} behaves like the simulator S_{sub} from the work of Kuhn et al. [60] except that acts on one kind of message differently to S_{sub}: if an $\mathcal{F}_{SCS}$ instance sends a message $p := (\texttt{"sent"}, P_i, Map, \mathsf{size})$, where "sent" is a string, P_i is the sender of a packet, Map is the next funnel in the protocol, size is the size of a packet. In that case, S^*_{sub} sends the message p directly to A_d, which is running inside S_{sub}. As $\mathcal{A}_d$ is stateless, these extra messages do not change $\mathcal{A}_d$'s behaviour.*

The protocol Streams *still follows the code in Figure 5.6 and Fig. 5.7, except that it called* $\mathcal{F}_{sub}$ *instead of* Π_{sub}.

Claim 10. *There is a simulator* S_{sub} *such that Game 1 is indistinguishable from Game 2.*

Proof of Claim A. nalogously to UC's completeness theorem, it suffices to consider the dummy attacker $\mathcal{A}_d$ that forwards all messages to the environment and only acts on the environment's orders.[5] We replace $\mathcal{A}_d$ with a simulator S^*_{sub} that almost behaves like the simulator S_{sub} in the work of Kuhn et al. [60], which internally runs $\mathcal{A}_d$. S^*_{sub}, however, has to also present an indistinguishable view for $\mathcal{E}$; hence, it has to forward all $\mathcal{F}_{SCS}$ notifications to the environment, just as $\mathcal{A}_d$ would do.

Next, we show that Game 1 (running Π_{sub}, $\mathcal{F}_{RKR}$, and $\mathcal{A}_d$) and Game 2 (running $\mathcal{F}_{sub}$ and S^*_{sub}) are indistinguishable. We show that, if Game 1 is distinguishable from Game 2, then Π_{sub} does not UC realize $\mathcal{F}_{sub}$, which contradicts [60]. $\mathcal{F}_{RKR}$ is faithfully simulated within S_{sub}; hence, it behaves exactly the same in these two interactions.

Towards contradiction, assume that Game 1 is distinguishable from Game 2. Given an environment $\mathcal{E}$ that can distinguish Game 1 from Game 2, we construct an environment $\mathcal{E}_{sub}$ that can distinguish Π_{sub} and $\mathcal{F}_{RKR}$ interacting with the dummy attacker $\mathcal{A}_d$ from $\mathcal{F}_{sub}$ interacting with S_{sub}. $\mathcal{E}_{sub}$ internally runs $\mathcal{E}$. $\mathcal{E}_{sub}$ has to ensure that $\mathcal{E}$ believes that it is in Game 1 or Game 2, respectively. Hence, $\mathcal{E}_{sub}$ has to ensure that $\mathcal{E}$ gets the same messages as in Game 1 and Game 2, respectively. So, we have to make sure that $\mathcal{E}$ sees the same the funnel-protocol communication and the notification messages from $\mathcal{F}_{SCS}$ for each packet that are handed through from $\mathcal{A}_d$ in Game 1. The funnel-protocol communication can be achieved by $\mathcal{E}_{sub}$ running the funnel-protocol instances. In Game 1 and Game 2, the $\mathcal{F}_{SCS}$ notification messages are sent whenever a funnel or a compute node instance communicates with $\mathcal{F}_{round}$. Hence, $\mathcal{E}_{sub}$ has to ensure that $\mathcal{E}$ gets these notification messages at the correct time, which can do as it internally runs $\mathcal{F}_{SCS}$.

- **The environment** $\mathcal{E}_{sub}$ internally runs $\mathcal{F}_{SCS}$, $\mathcal{F}_{round}$, and $\mathcal{E}$. Let $Int_{1,sub}$ be the interaction between Π_{sub} and $\mathcal{F}_{RKR}$ from [60] and the dummy attacker $\mathcal{A}_d$ with an

[5]↑For any other attacker $\mathcal{A}$ and each environment $\mathcal{E}$, there is an environment $\mathcal{E}$ that internally emulates the interaction between $\mathcal{E}$ and $\mathcal{A}$. $\mathcal{E}$ interacts with the dummy attacker $\mathcal{A}_d$ and produces the same view.

environment (in our case $\mathcal{E}_{sub}$), and let $Int_{2,sub}$ be the interaction between $\mathcal{F}_{sub}$ and the simulator S_{sub} from [60]. As $\mathcal{E}_{sub}$ does not know whether it is interacting with $Int_{1,sub}$ or $Int_{2,sub}$, we describe its behavior agnostic to $b = 1$ or $b = 2$ with $Int_{b,sub}$.

- Upon receiving a message a party from $Int_{b,sub}$ (i.e., from a Π_{sub} instance or $\mathcal{F}_{sub}$), run Π_{client} and forward the response to $\mathcal{E}$.
- Upon receiving a message over the network from $Int_{b,sub}$ (i.e., from $\mathcal{A}_d$ or S_{sub}), forward the message to $\mathcal{F}_{SCS}$ and faithfully (as in Game 1) compute the interaction between $\mathcal{F}_{round}$, the funnel instances, and $\mathcal{E}$.
- Upon receiving a message from $\mathcal{E}$ for the network attacker, directly forward this message to the network attacker in $Int_{b,sub}$ (i.e., to $\mathcal{A}_d$ or $\mathtt{S}_{sub}$).
  - Upon receiving a notification message from the internally emulated $\mathcal{F}_{SCS}$, forward it to $\mathcal{E}$. (We stress that S^*_{sub} is split into this interaction and the part that is run in Int_{sub}.)

For each $b \in \{1, 2\}$, we have to show that for $\mathcal{E}$ the interaction within $\mathcal{E}_{sub}$, which in turn interacts with $Int_{b,sub}$, is indistinguishable from the interaction with Game b. For $b = 1$, the interaction within $\mathcal{E}_{sub}$ solely differs in the order in which the notification message from $\mathcal{F}_{SCS}$ arrives. As these messages first reach $\mathcal{E}_{sub}$ before reaching $\mathcal{E}$, $\mathcal{E}_{sub}$ can successfully reverse the order again (see above) and constructs a perfect view for $\mathcal{E}$.

For $b = 2$, $\mathcal{E}_{sub}$ internally emulates Game 1 (except for Π_{sub}). We show that for $b = 2$ nevertheless the view of $\mathcal{E}$ when being emulated within $\mathcal{E}_{sub}$ is indistinguishable from the view when interacting in Game 2. Recall that the only difference between Game 1 and Game 2 is that Π_{sub} is replaced by $\mathcal{F}_{sub}$, and $\mathcal{A}_d$ is replaced by $\mathtt{S}_{sub}$. As $\mathcal{F}_{sub}$ is changed by $\mathcal{E}_{sub}$, it suffices to analyze whether the message transcript to $\mathtt{S}_{sub}$ is indistinguishable for $\mathtt{S}_{sub}$ and whether the transcript from $\mathtt{S}_{sub}$ (through $\mathcal{E}_{sub}$) is indistinguishable for $\mathcal{E}$.

Whenever by $Int_{2,sub}$ a message is sent by $\mathtt{S}_{sub}$ to $\mathcal{E}_{sub}$, this messages first goes through the internally emulated instances of the $\mathcal{F}_{SCS}, \mathcal{F}_{round}$, and Π_{funnel} protocols. These protocols solely forward messages, and of these only $\mathcal{F}_{SCS}$ sends a notification to the network attacker $\mathcal{A}_d$. In this case, as defined above, $\mathcal{E}_{sub}$ directly forwards the notification to $\mathcal{E}$; this is exactly

the same that would happen in Game 2. All other messages are forwarded and, as in Game 2, potentially sent to $\mathcal{E}$. Hence, whenever in Game 2 a message is sent to $\mathcal{E}$ also in $\mathcal{E}_{sub}$'s internal emulation (if $b = 2$) a message is sent to $\mathcal{E}$.

Next, we consider the case where for $b = 2$ a message is sent by $\mathcal{E}$ (while it is being internally emulated by $\mathcal{E}_{sub}$) to the network attacker, which would in Game 2 be $\mathtt{S}^*_{sub}$. As defined above, in this case, $\mathcal{E}_{sub}$ sends the message directly to the network attacker in $Int_{b,sub}$. As $b = 2$, the network attacker is $\mathtt{S}_{sub}$. Hence, the message transcript (from $\mathtt{S}_{sub}$'s point of view) is exactly the same as in Game 2.

If Int_{sub} is the interaction with Π_{sub}, $\mathcal{F}_{RKR}$, and $\mathcal{A}_d$, $\mathcal{E}_{sub}$ ensures that $\mathcal{E}$ has exactly the same view as in Game 1. If Int_{sub} is the interaction with $\mathcal{F}_{sub}$ and S_{sub}, $\mathcal{E}_{sub}$ ensures that $\mathcal{E}$ has exactly the same view as in Game 2. Hence, by assumption, with the translation of $\mathcal{E}_{sub}$ the submachine $\mathcal{E}$ can distinguish the interaction with Π_{sub}, $\mathcal{F}_{RKR}$, and $\mathcal{A}_d$ from the interaction with $\mathcal{F}_{sub}$ and S_{sub}.

For any poly-bounded $\mathcal{E}$, $\mathcal{E}_{sub}$ acts as a poly-bounded environment in the UC game. Yet, Kuhn et al. [60] proved that there is no poly-bounded environment that can distinguish these two interactions, which is a contradiction. Hence, Game 1 and Game 2 are indistinguishable.

◇

Game 3. *We replace Π_{worker}, Π_{client}, $\mathcal{F}_{SCS}$, $\mathcal{F}_{sub}$ with the ideal functionality $\mathcal{F}_{\mathsf{Streams}}$. The simulator S^*_{sub} is replaced by a simulator S_f. The simulator S_f internally runs S^*_{sub} but translates the format of the output of $\mathcal{F}_{\mathsf{Streams}}$ to the format output by $\mathcal{F}_{sub}$. We stress that as we are in the hybrid $\mathcal{F}_{round}$-model, $\mathcal{F}_{round}$ remains in the ideal world as it was in the previous games.*

Claim 11. *With the simulator S_f, Game 3 is indistinguishable from Game 2.*

Proof of Claim F. or the analysis, we divide the execution in overlapping sub-sequences of the form compute node, funnel, compute node (overlapping at the last funnel). For those sub-sequences where the funnel is malicious or the first compute node is malicious, $\mathcal{F}_{\mathsf{Streams}}$ has exactly the same leakage as $\mathcal{F}_{sub}$, except that the format of the leakage is translated. If one of the funnels is honest and the first compute nodes is honest, though, $\mathcal{F}_{\mathsf{Streams}}$, in

contrast to $\mathcal{F}_{sub}$, does not leak which compute node sends (the ideal abstraction of) an onion to which other compute node. Next, we argue that this leakage is also hidden in Game 2, as $\mathcal{F}_{SCS}$ and the funnels hide this information.

As the first compute node is honest, it does not leak to the network attacker to whom the onion is sent. If the first funnel is honest, it does not leak the link between the two compute nodes to the network attacker. As $\mathcal{F}_{SCS}$ only notifies the network attacker that some messages was sent and as the funnels shuffle the messages of each round, the network attacker does not learn by whom an onion was sent. ◇

Therefore, for the simulator $\mathcal{S}$ our protocol Streams UC-realizes the ideal functionality $\mathcal{F}_{\mathsf{Streams}}$ in the $\mathcal{F}_{round}$-hybrid model. □

5.A.2 Security Analysis Proofs

Theorem 5.3.1. If the amount of compromised nodes is a constant fraction $\frac{\mathsf{c}}{\mathsf{K}} < 1$, $\mathcal{F}_{\mathsf{Streams}}$ provides pairwise unlinkability of messages over $\mathfrak{L}$ rounds up to probability δ as in Definition 5.1.1, where $\delta < \gamma^{\mathfrak{L}}$ for some constant constant fraction $0 < \gamma < 1$.

Proof. Recall that, we assume that each node in a round is chosen uniformly at random (funnel nodes by randomness beacon and compute nodes by the clients) with replacement, and independent of all other rounds. Conceptually, a careful strategy where nodes are chosen by avoiding repetition as much as possible can provide better security guarantees. For the ease of analysis, we make such weaker assumption.

If two messages remain in $\mathcal{F}_{\mathsf{Streams}}$ for $\mathfrak{L}$ rounds, they are shuffled if both of those two messages have honest compute nodes on their path in some round r, and then an honest node is picked as the funnel node in round $r + 1$. If that happens, a shuffled list of newly generated temporary identifiers are given to $\mathcal{S}$ on behalf of those messages. In Figure 5.8, we pictorially show the possible cases when two messages can mix (or not).

Let a be the probability of a randomly picked node being honest; $a = \frac{\mathsf{K}-\mathsf{c}}{\mathsf{K}}$ Since the funnel node is picked uniformly at random (with replacement, and independent of all other nodes), the probability of the funnel node being compromised is $\frac{\mathsf{c}}{\mathsf{K}} = (1 - a)$, and being honest is a. Similarly, each compute node on the path of a message is selected uniformly at

random (with replacement, and independent of all other nodes). Therefore, the probability of an compute node being honest is also a.

Therefore, the probability that the two messages mix in for a given pair of compute node in round r and funnel node in round $(r+1)$ is a^3. And, the probability that they don't mix in those pair of rounds is $Y = 1 - a^3$. If two messages stay in the system together for $\mathfrak{L}$ rounds, they do not mix with probability at most

$$\delta < \left(1 - a^3\right)^{\frac{\mathfrak{L}}{2}}$$

Y is conceptually the proportion of supernodes where the two messages cannot mix. In Figure 5.11a we plot the relationship between Y and $\frac{\mathsf{c}}{\mathsf{K}}$. For a constant $\frac{\mathsf{c}}{\mathsf{K}}$, Y is constant; and hence $\delta < \gamma^{\mathfrak{L}}$ where $\gamma = (1 - a^3)^{1/2}$ □

Theorem 5.4.3. Given a constant fraction $\frac{\mathsf{c}}{\mathsf{K}}$, against any adversary $\mathcal{S}$, if two messages stay together for $\mathfrak{L} \in \omega(\log \eta)$ rounds in the core protocol described in Section 5.2.3 they are shuffled with a probability $(1 - \delta)$, where $\delta < \left(\frac{\mathsf{c}}{\mathsf{K}}\right)^{\mathfrak{L}}$.

Proof Sketch. We skip the detailed proof as the proof methodology is very similar to the proof with supernodes. The UC proof becomes much easier if we do not have to distinguish between compute and funnel nodes. And for the combinatorial argument there is one key difference: instead of the funnel node in the funnel round and the two compute nodes in the immediate next compute round, there is only one node and just one round.

Therefore, instead of representing each pair of rounds with three coin tosses in the supernode scenario, we have exactly one coin toss per round for the core protocol with success probability $a = \frac{\mathsf{c}}{\mathsf{K}}$. And hence, we can define an ideal functionality $\mathcal{F}_{core}$ as described in Fig. 5.15, which shuffles the messages in the system whenever they encounter an honest node on the path. To provide a simulator for $\mathcal{F}_{core}$ we use the exact same simulator $\mathcal{S}_{sub}$ as the one we use in the proof of Theorem 5.4.1, with only one minor modification that $\mathcal{S}_{sub}$ directly forwards all the network messages to the round functionality.

From the ideal functionality it is simple to show that the probability of **not** finding an honest node in a path of length $\mathfrak{L}$ is upper bounded by $\delta \leq \left(\frac{\mathsf{c}}{\mathsf{K}}\right)^{\mathfrak{L}}$. Therefore, if two arbitrary

messages stay in the protocol for at least $\mathfrak{L}$ rounds, they are shuffled with probability at least $1 - \delta$. □

5.B Existing functionalities

Ideal Functionality for Onion Routing. We borrow the ideal functionality $\mathcal{F}_{or}$ for onion routing from the work of Kuhn et al. [60, Algorithm 1]. We present the ideal functionality in Figure 5.16 for completeness. Kuhn et al. [60, Appendix E] also presents a modified version of Sphinx [78] that realizes the ideal functionality $\mathcal{F}_{or}$. We use the same modified version of Sphinx as our Π_{or} in the current work.

Secure Communications Sessions. We use the ideal functionality $\mathcal{F}_{scs}$ for secure communications sessions from the work of Gajek et al. [76, Figure 4]. Their work shows that TLS protocol [76, Figure 5] UC realizes the ideal functionality $\mathcal{F}_{scs}$.

inputBuffer[] an array of queues to store messages for nodes
crf = an infinitely long random string
dlvrCM, *dlvrHM*, *queue*, DB are hashmaps.
$\texttt{round} := 0$, $newRound[] := \{\texttt{false}, \texttt{false}, \dots\}$
$partyCount := 0$

Upon new round from $\mathcal{E}$ for party P:

if $newRound(P) = \textbf{true}$ **then**
  return "invalid action"
set $newRound(P) := \textbf{true}$; $partyCount += 1$
if P is a client **then**
  $(m, R, t) \leftarrow$ dequeue $inputBuffer[P]$
  $d \leftarrow DelayDistribution(\frac{\ell}{2}, \ell - 1)$;
  $\{x_1, \dots, x_d\} \xleftarrow{\$} \{\mathsf{crf}[\texttt{round}]\%\mathsf{K}, \dots, \mathsf{crf}[\texttt{round} + d]\%\mathsf{K}\}^d$
  if $!\exists x \in \{x_1, \dots, x_d\}$ such that $x_a \in \mathtt{I_h}$ **then**
    Send $(m, x_1, \dots, x_d)$ to $\mathcal{S}$
  else
    let $x_a :=$ the **first** honest party on the path $\{P, x_1, \dots, x_d\}$
    Send $(q, x_1, \dots, x_a)$ to $\mathcal{S}$ where $q \xleftarrow{\$} \mathcal{M}$
    store $(q, x_a, m, x_{a+1}, \dots, x_d, R)$ in $queue(\texttt{round} + a)$
if $partyCount = \mathsf{N} + \mathsf{K}$ **then**
  SendInformation()
NextRound(P)

Upon input message (m, R, t) from $\mathcal{E}$ for party P:

$inputBuffer(P) += 1$
if $\texttt{round} \neq t$ **then**
  reject packet and exit
Add (m, R, t) in $inputBuffer[P]$

Upon receiving a message m for party P:

Output "Message m received" to $\mathcal{E}$

`SendInformation`()

for each $(q, x_a, m, x_{a+1}, \dots, x_d, R) \in queue(\texttt{round})$ **do**
  Remove $(q, x_a, m, x_{a+1}, \dots, x_d, R)$ from $queue(\texttt{round})$
  let x_γ be the **next** honest node on the path $\{x_{a+1}, \dots, x_d\}$
  if there is **no** such x_γ **then**
    Add $(m, x_{a+1}, \dots, x_d, R)$ in a temporary queue Q
  else
    Add $(q, x_{a+1}, \dots, x_\gamma)$ in Q for $q \xleftarrow{\$} \mathcal{M}$
    Add $(q, x_\gamma, m, x_{\gamma+1}, \dots, x_d, R)$ to $queue(\texttt{round} + \gamma - a)$
Shuffle the elements of Q and send them to $\mathcal{S}$

Figure 5.15. Ideal functionality $\mathcal{F}_{core}$ for the Core Protocol

Data structure:
Bad: Set of Corrupted Nodes
L: List of Onions processed by adversarial nodes
B_i: List of Onions held by node P_i
// Notation:
// $\mathcal{S}$: Adversary (resp. Simulator)
// $\mathcal{Z}$: Environment
// $\mathcal{P} = (P_{o_1}, \ldots, P_{o_n})$: Onion path
// $O = (sid, P_s, P_r, m, n, \mathcal{P}, i)$: Onion = (session ID, sender, receiver, message, path length, path, traveled distance)
// N: Maximal onion path length

On message Process_New_Onion$(P_r, m, n, \mathcal{P})$ *from* P_s
- // P_s creates and sends a new onion (either instructed by $\mathcal{Z}$ if honest or $\mathcal{S}$ if corrupted)
- **if** $|\mathcal{P}| > N$; // selected path too long **then**
  - Reject
- **else**
  - $sid \leftarrow^R$ session ID ; // pick random session ID
  - $O \leftarrow (sid, P_s, P_r, m, n, \mathcal{P}, 0)$; // create new onion
  - Output_Corrupt_Sender$(P_s, sid, P_r, m, n, \mathcal{P}, \text{start})$ Process_Next_Step(O)

Procedure Output_Corrupt_Sender$(P_s, sid, P_r, m, n, \mathcal{P}, temp)$
- // Give all information about onion to adversary if sender is corrupt
- **if** $P_s \in Bad$ **then**
  - Send "$temp$ belongs to onion from P_s with $sid, P_r, m, n, \mathcal{P}$" to $\mathcal{S}$

Procedure Process_Next_Step$(O = (sid, P_s, P_r, m, n, \mathcal{P}, i))$
- // Router P_{o_i} just processed O that is now passed to router $P_{o_{i+1}}$
- **if** $P_{o_j} \in Bad$ *for all* $j > i$ **then**
  - Send "Onion from P_{o_i} with message m for P_r routed through $(P_{o_{i+1}}, \ldots, P_{o_n})$" to $\mathcal{S}$
  - Output_Corrupt_Sender$(P_s, sid, P_r, m, n, \mathcal{P}, \text{end})$
- **else**
  - // there exists an honest successor P_{o_j}
  - $P_{o_j} \leftarrow P_{o_k}$ with smallest k such that $P_{o_k} \notin$ Bad $temp \leftarrow^R$ temporary ID Send "Onion $temp$ from P_{o_i} routed through $(P_{o_{i+1}}, \ldots, P_{o_{j-1}})$ to P_{o_j}" to $\mathcal{S}$
  - Output_Corrupt_Sender$(P_s, sid, P_r, m, n, \mathcal{P}, temp)$ Add $(temp, O, j)$ to L

On message Deliver_Message$(temp)$ *from* $\mathcal{S}$
- // Adversary $\mathcal{S}$ (controlling all links) delivers onion belonging to $temp$ to next node
- **if** $(temp, \_, \_) \in L$ **then**
  - Retrieve $(temp, O = (sid, P_s, P_r, m, n, \mathcal{P}, i), j)$ from L $O \leftarrow (sid, P_s, P_r, m, n, \mathcal{P}, j)$**if** $j < n + 1$ **then**
    - $temp \leftarrow^R$ temporary ID Send "$temp$ received" to P_{o_j} Store $(temp, O)$ in B_{o_j}
  - **else**
    - **if** $m \neq \perp$ **then**
      - Send "Message m received" to P_r

On message Forward_Onion$(temp)$ *from* P_i
- // P_i is done processing onion with $temp$ (either decided by $\mathcal{Z}$ if honest or $\mathcal{S}$ if corrupted)
- **if** $(temp, \_) \in B_i$ **then**
  - Retrieve $(temp, O)$ from B_i Remove $(temp, O)$ from B_i Process_Next_Step(O)

Figure 5.16. Ideal Functionality $\mathcal{F}_{or}$ from [60].

6. IMPLICATIONS OF TRILEMMA RESULTS: CONSTRUCTING AC PROTOCOL AT THE COST OF BANDWIDTH OVERHEAD

In the last chapter we constructed a mixnet-based protocol *Streams* that provides provable mixing property at the expense of latency overhead. However, Streams did not use any user coordination technique. In this chapter we are going to design a DC-net based protocol OrgAn that effectively utilizes user coordination to achieve anonymity. In this process we provide a novel technique of user coordination that has significantly less overhead compared to existing DC-net based protocols.

With our design we solve three key problems that exists in most standard DC-net designs: (i) First, all the users need to run a key agreement protocol among themselves to agree on shared secrets keys; such an agreement protocol is not scalable as it comes with high communication overhead and has to be repeated often towards stopping linkability/co-relations across multiple rounds. (ii)Second, it requires all the users to participate in a slot agreement protocol before every round; otherwise two or more messages may collide as only one user is supposed to send a message in any given round. (iii) Finally, the DC-net designs draws their efficiency gains over mixnets through the fixed user setup and co-ordination between them [83]: Unlike for mixnets, any user arrival, absentees and departure mandates re-running the setup with the new group.

Solutions such as Dissent in Numbers [5], Verdict [37], MCMix [53] mitigate the first scalability problem by shifting the key agreement to the second tier: here, a set of servers perform DC-net like protocol on users' behalf and privacy is maintained as long as any one of the servers remains honest. (This is known as the *anytrust* assumption.) DiceMix [15] avoids the second slot agreement problem employing Newton's power sums formulation [84] by allowing all the n users to send one message each in a single round of the protocol. PowerMix [58] combines both ideas with the multi-party computation (MPC) as a service towards overcoming the above three problems, and recently, Blinder [59] protocol made the server-enabled design more efficient for a reduced adversarial threshold. To minimize the risks of coercion and collusion, these servers should be typically spread across different

geo-political jurisdictions, and these designs introduce significant latency overhead as the underlying MPC still require significant interaction among the servers.

We solve the above problems in an organizational setting, utilizing a client/relay/server architecture introduced by Barman et al. [36]. In this architecture, a set of few servers help the clients in establishing shared secrets among themselves, but actual anonymous communication happen through another relay node.

6.1 System Overview And Protocol Idea

In this section we provide an overview of system setup, security and system goals, and a brief solution overview.

6.1.1 Setup and Communication Model

Consider an organizational network with N clients $u_1, \ldots, u_{\mathsf{N}}$. They wish to access intra-organizational services as well as connect to services outside the organizational network without revealing which client is actually communicating. For this, they use our protocol OrgAn. Other than the clients, the infrastructure for OrgAn consists of a set of K setup servers denoted as $G_1, \ldots, G_{\mathsf{K}}$, and one relay node R. Using the relay node and the setup servers, the clients want to achieve anonymity while communicating with different services.

The relay R is a gateway server between the organizational network and the outside world. It helps the clients of the OrgAn protocol to transmit messages outside as well as to intra-organization services, but does not act as a trusted third party in the anonymization process. All the messages from the clients of the system (outbound traffic) are transmitted through the relay to the services (intra-oraganization or outside world). All the response messages from the outside world (inbound traffic) are received by the relay and forwarded to the clients. We assume that the relay has high availability and high computation power.

Additionally there is a small set of K servers $G_1, \ldots, G_{\mathsf{K}}$ outside the organization, we call them setup servers. These setup servers help the clients in the setup or key agreement process, but do not take part in the anonymization process. These servers could be maintained by

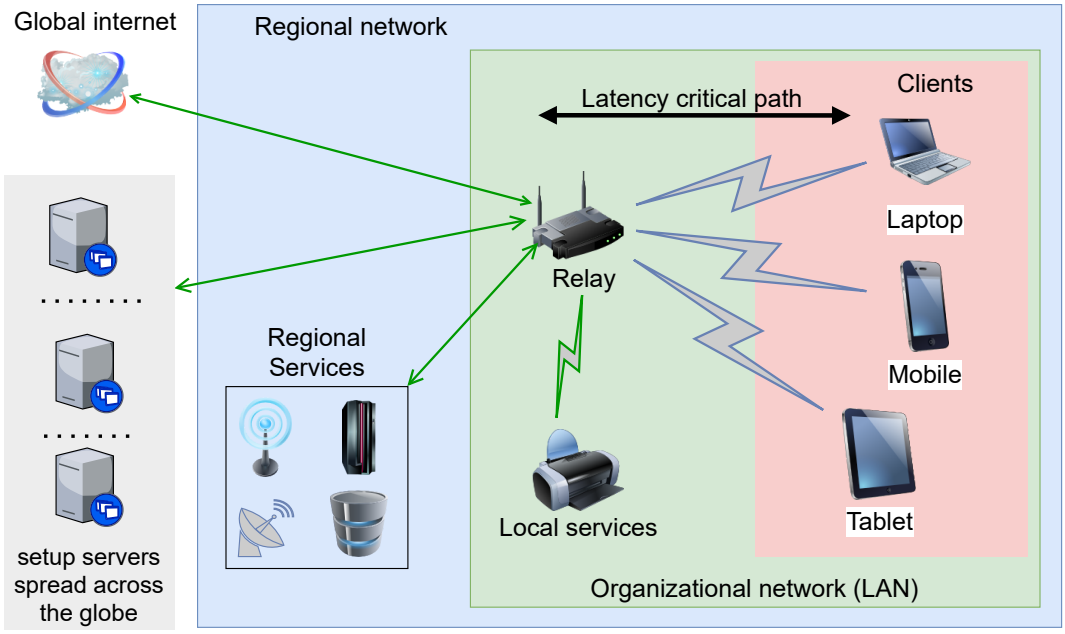


Figure 6.1. System overview of OrgAn

independent third parties outside the organization and are assumed to be scattered across the globe.

We do not require the setup servers and the relay to communicate with each other during the setup phase or protocol run — they do not even need to know each other. The clients can mutually agree on the set of setup servers, ensuring that for each client there is at least one setup server that they trust. We do not require the setup servers to be online except for the setup phase.

In this work we only focus on the anonymity of outbound traffic (sender anonymity). We consider the solution provided by PriFi for inbound traffic (recipient anonymity) adequate and can be easily fit into our protocol; hence, we do not consider inbound traffic for the rest of the chapter.

6.1.2 Threat Model

We consider a probabilistic polynomial time (PPT) adversary $\mathcal{A}$ who can observe all network traffic. Additionally, the adversary can compromise some clients. However, we

assume that at least two clients are honest. We assume that at least one of the setup servers is honest.

Moreover, the relay can be under adversarial control, but we do not consider denial-of-service (DoS) attacks from the relay, because the clients can easily identify if the availability of the relay is compromised — and the organization does not want to openly show that they are against privacy of their members/employees.

Since we focus on demonstrating the applicability of our trilemma results in the context of provable security against global passive adversaries, we do not discuss here about active attacks and the defenses against them, and leave those for future work.

6.1.3 Goals

Anonymity. We want our protocol to achieve *sender anonymity* for the outbound traffic, i.e., the (in-)ability of any third party or the relay to figure out which user has sent a specific message, even if all but two clients and all but one server (setup node) are compromised. We want *strong* sender anonymity, which means, the adversarial advantage in guessing the sender correctly is at most negligibly better than random guessing.

Low Latency. The system should operate with low latency overhead to support applications like voice and video calls.

Reduced Bandwidth Overhead. We want to minimize (preferably eliminate completely) any communication overhead because of the setup servers during the protocol run. We also want to minimize the communication overhead among the clients and the relay node.

Scalability. We want OrgAn to support small to medium organizations (i.e., up to several hundred clients). Inherently user coordination techniques cannot scale for hundreds of thousands of users — it will be an interesting future work to come up with a user coordination technique that scales gracefully with number of users.

6.1.4 Protocol Idea

Our protocol design is based on the idea that each client u_i has an individual secret $\mathbf{r}_i$ — here $\sum_i \mathbf{r}_i$ is known to everyone, but the individual values of $\mathbf{r}_i$ values with the honest

clients are not known. The clients achieve that using a setup phase, with the help of a few setup servers. If there is at least one honest setup node, this setup can be achieved by a secret sharing scheme between the setup servers and the clients.

In our protocol design, we use a key-homomorphic PRF $\mathcal{F}$ that satisfies $\mathcal{F}(\mathbf{k}, x) = \mathcal{F}(\mathbf{k}_1, x) + \mathcal{F}(\mathbf{k}_2, x)$. In a round d, each client u_i uses an element of $\mathcal{F}(\mathbf{r}_i, d)_t$ as the mask for their DC-net cipher that they send to the relay for a slot t. Since the relay knows $\sum_i \mathbf{r}_i$ but not the individual $\mathbf{r}_i$ values associated with the honest clients, it can decrypt the message for slot t only after receiving the DC-net ciphers from all the clients for that slot by computing $\mathcal{F}(\sum_i \mathbf{r}_i, d)_t$. We solve the slot agreement problem generally faced by DC-net based protocols, by constructing powersums of messages similar to Dicemix [15]. However, we do it in less number of rounds than Dicemix by exploiting the network model and the key homomorphic property of $\mathcal{F}$. We detail our protocol description in Section 6.3.

6.2 Building Blocks

Power-sum equations and solution [15,84]. Consider the following system of equations:

$$
\begin{aligned}
E(1) &= x_1 + x_2 + \cdots + x_{\mathsf{N}} \\
E(2) &= x_1^2 + x_2^2 + \cdots + x_{\mathsf{N}}^2 \\
&\cdots \\
E(\mathsf{N}) &= x_1^{\mathsf{N}} + x_2^{\mathsf{N}} + \cdots + x_{\mathsf{N}}^{\mathsf{N}}
\end{aligned}
$$

with each $x_i \in \mathbb{Z}_{\mathsf{p}}$. where all the elements x_i, their powers, and the sums of their powers are elements in a finite field $\mathbb{F}$. This equation system can be solved using Newton's identities [15, 84]. Mathematically we denote the function as $\mathsf{SolveEqn}(E(1), \ldots, E(\mathsf{N}))$ that takes such an equation system as input, and outputs an unordered set of N elements $\{x_1, x_2, \ldots, x_{\mathsf{N}}\}$, if the equation system is solvable. In our protocol, each x_i is the input of client u_i; the equation system is computed and solved by the relay R using the $\mathsf{SolveEqn}()$ function to find an unordered set of client messages x_i.

Key-Homomorphic Pseudorandom Function. A key-homomorphic pseudo random function family (PRF) [85] is a PRF which is homomorphic in the key-input of the function, i.e., $\mathcal{F}(\mathbf{k}, x) = \mathcal{F}(\mathbf{k}_1, x) + \mathcal{F}(\mathbf{k}_2, x)$.

Additionally, in our protocol we use a digital signature scheme [86] that is existentially unforgeable under chosen-message attacks [87]. Let $(\mathcal{S}, \mathcal{V})$ be the signature scheme — given a private-public key pair (p, P), $\sigma = \mathcal{S}_p(x)$ denotes the signature of message x with the key p, and using $\mathcal{V}_P(x, \sigma)$ anyone can verify the signature.

6.3 Protocol Description

In this section we present the OrgAn protocol. As part of this work, since we are interested in demonstrating an effective method of user coordination against global passive adversaries, we do not consider any active attacks.

As mentioned earlier in Section 6.1.1, our system consists of the following set of parties:

- a set of N clients denoted as $u_1, \ldots, u_{\mathsf{N}}$ that (or some of them) want to communicate with the outside world;
- a set of K setup servers denoted as $G_1, \ldots, G_{\mathsf{K}}$ the reside outside the organizational network;
- one relay server R that acts as a gateway.

We assume that all the protocol parties in our system have access to a public key infrastructure (PKI), where each party X has a long term private-public key pair (p_X, P_X).

Our protocol first runs a one-time setup phase, where the clients receive random secret-shares of a value known to relay from the setup servers, and then start running the protocol. In our core protocol (without active attacks), the clients need to run the setup phase only once, and never again.

6.3.1 Setup Phase

In the setup phase, each setup server G_j splits a publicly known value $\mathbf{s}$ into N secret shares $\{\mathbf{r}_{1j}, \ldots, \mathbf{r}_{\mathsf{N}j}\}$ and distributes the shares among the clients, where each client u_i receives the share $\mathbf{r}_{ij}$, such that $\sum_i \mathbf{r}_{ij} = \mathbf{s}$. All the $\mathbf{s}, \mathbf{r}_{ij}$ values and all the operations during

the protocol are in a finite field $\mathbb{F}$. Note here, all the setup servers use the same $\mathbf{s}$ value and that is a global parameter of the protocol, however, the shares for the clients generated by each honest setup server is independent of other servers, as well as unknown to other servers. We assume that the setup servers communicate with the clients using some authenticated and confidential channel (for example, using TLS [76,88]).

After receiving one share from each setup server, each client u_i has the following secrets: $\{\mathbf{r}_{i1}, \ldots, \mathbf{r}_{i,\mathsf{K}}\}; \mathbf{r}_{ij}$. Each client u_i computes the following.

$$\mathbf{r}_i = \sum_{j=1}^{m} \mathbf{r}_{ij} \tag{6.1}$$

We present the pseudo-code for the setup run by the setup servers and the clients in Figure 6.2.

Remark 1. *The adversary always knows that* $\sum_i \mathbf{r}_{ij} = \mathbf{s}$. *If there are two honest clients* u_1 *and* u_2, *the adversary can always compute* $\mathbf{r}_{1j} + \mathbf{r}_{2j}$ *for any honest setup server* j. *That is the only leakage after the setup phase — the adversary cannot guess the individual values of* $\mathbf{r}_{1j}$ *and* $\mathbf{r}_{2j}$.

$\mathbf{s} \in \mathbb{F}$; a global system-parameter

ServerSetup (setup server G_j, set $\mathcal{S} = \{u_1, \ldots, u_{\mathsf{N}}\}$) :

$\{\mathbf{r}_{1j}, \ldots, \mathbf{r}_{\mathsf{N}j}\}$ = split $\mathbf{s}$ into N shares
Send $\mathbf{r}_{ij}$ to user u_i over TLS for each $i \in \{1, \ldots, \mathsf{N}\}$

ClientSetup (user u_i, set $I = \{G_1, \ldots, G_{\mathsf{K}}\}$) :

$\mathbf{r}_{i1}, \ldots, \mathbf{r}_{i\mathsf{K}}$ = Wait for shares from each $G_j \in I$
$\mathbf{r}_i = \sum_{j=1}^{\mathsf{K}} \mathbf{r}_{ij}$
Send "Setup completed" to the relay R

RelaySetup() :

Initiate **ServerSetup**$(G_j, \mathcal{S})$ for each $G_j \in I$
Initiate **ClientSetup**(u_i, I) for each $u_i \in \mathcal{S}$
Wait for "Setup completed" from each $u_i \in \mathcal{S}$

Figure 6.2. Setup protocol in OrgAn

6.3.2 Protocol Run

Our protocol can be divided into several *rounds*; in one round each client u_i can send one message x_i. In every round, the relay R maintains N slots to receive N equations. The relay retrieves the N messages from N clients by solving those equations. For each round d, the protocol is run in the following steps:

Client Ciphertext generation. Each client u_i with a message $x_i \in \mathbb{F}$ computes the following, for each slot t in the round d:

$$p_i(t) = \mathcal{F}\left(\mathbf{r}_i, d \times \mathsf{N} + t\right)$$
$$c_i(t) = x_i^t + p_i(t)$$

Sender u_i then sends the ordered set $\{c_i(1), \ldots, c_i(\mathsf{N})\}$ tagged with the round number d to the relay.

Slot value reveal. For a slot t, the relay R collects the ciphertexts $c_1(t), \ldots, c_{\mathsf{N}}(t)$ from all the users and computes the following:

$$\begin{aligned} P(t) =& c_1(t) + \cdots + c_{\mathsf{N}}(t) - \mathcal{F}(\mathsf{K} \cdot \mathbf{s}, d \times \mathsf{N} + t) \\ =& x_1^t + x_2^t + \cdots + x_{\mathsf{N}}^t = E(t) \end{aligned}$$

Once all the slot values from all the clients are received, the relay has $E(1), \ldots, E(\mathsf{N})$. The relay can solve the above equation system to retrieve $x_1, x_2, \ldots, x_{\mathsf{N}}$ (without knowing which message belongs to which client) using $\mathsf{SolveEqn}(E(1), \ldots, E(\mathsf{N}))$. Once the individual values $x_1, x_2, \ldots, x_{\mathsf{N}}$ are retrieved, the relay can forward them to the outside world. We present the pseudocode for the protocol in Figure 6.3. The setup servers do not take part during the protocol run at all.

Remark 2. *$E(1) = x_1 + x_2 + \cdots + x_{\mathsf{N}}$ can be seen by the adversary in plain text. If all but two clients (without loss of generality let us assume u_1 and u_2) are compromised, the adversary knows $(x_1 + x_2)$. However, this leakage does not provide any additional information. Because, after solving the equation system the relay anyway knows the values.*

hs: a global system-parameter
T: number of slots to be used in the current round

RelayProtocol(round d):

$P_1, \ldots, P_T = -\mathcal{F}(\mathsf{K} \cdot \mathbf{s}, d\mathsf{N} + t)$
for $i \in \{1, \ldots, \mathsf{N}\}$ **do**
  $(d_i, c_i(1), \cdots, c_i(T))$ = Wait for message from u_i
  for $t \in \{1, \ldots, T\}$ **do**
    $P_t = P_t + c_i(t)$
$x_1 \ldots, x_{\mathsf{N}} = \mathsf{SolveEqn}(P_1, \ldots, P_{\mathsf{N}})$
Broadcast $x_1, \ldots, x_{\mathsf{N}}$

ClientProtocol(client u_i, packet M, round d) :

x_i be the message that the client wants to send
for each $j \in \{1, \ldots, \mathsf{N}\}$ **do**
  $c_i(j) = \mathcal{F}(\mathbf{r}_i, d\mathsf{N} + j) + x_i^j$
Send $(d, c_i(1), \cdots, c_i(\mathsf{N}))$ to relay R

Figure 6.3. Protocol run in OrgAn

Multiple rounds. The value of $\mathcal{F}(\mathbf{r}_i, d \times \mathsf{N} + t)$ can be computed for arbitrarily large value of d. Which means, the protocol can be run for a large number of rounds without the need to rerun the setup. Additionally, the relay does not need to receive ciphertexts for different rounds in the correct sequence, the relay can map them correctly using the round tag and slot id associated with each ciphertext.

Although the round number d can be arbitrarily large in $\mathcal{F}(\mathbf{r}_i, d \times \mathsf{N} + t)$, for the purpose of forward secrecy we recommend running the setup once in every few days. That procedure at regular intervals can be invoked by the relay. If the relay does not run the setup regularly as expected, the clients will suspect the relay's malicious intentions.

6.4 Security Analysis

In this section, we prove the (sender) anonymity property of our protocol OrgAn in the passive adversary setting. We start with presenting the security definition of PRF that we use in our proofs — we borrow those definitions from existing literature [85, 89]. We use the sender anonymity definition defined in Section 2.1.

6.4.1 Security Definition for PRFs

Let $\mathsf{Rand}(\mathcal{D}, \mathcal{O})$ denotes the set of all functions with domain $\mathcal{D}$ and range $\mathcal{O}$. We consider a distinguisher $\mathcal{A}$ that tries to distinguish if a function g has been picked randomly from a given function family $\mathcal{F}$ or from $\mathsf{Rand}(\mathcal{D}, \mathcal{O})$, when $\mathcal{A}$ is given oracle access to g. We write $\mathcal{A}(g)$ to denote that $\mathcal{A}$ is given oracle access to g. We define the following security game:

Definition 6.4.1. *Let $\mathcal{F} : \mathcal{K} \times \mathcal{D} \to \mathcal{O}$ be a family of efficient functions, and let $\mathcal{A}$ be an algorithm the takes an oracle for a function and returns a bit b. We consider the following two experiments:*

$\mathbf{Expt}_{PRF}(\mathcal{A})$	$\mathbf{Expt}_{g}(\mathcal{A})$
$\mathbf{K} \leftarrow \mathcal{K}$	$g \leftarrow \mathsf{Rand}(\mathcal{D}, \mathcal{O})$
$b = \mathcal{A}(\mathcal{F}_{\mathbf{K}})$	$b = \mathcal{A}(g)$

The adversarial advantage of $\mathcal{A}$ is defined as $\mathbf{Adv}_{\mathcal{F}}(\mathcal{A}) = \Pr[\mathbf{Expt}_{PRF}(\mathcal{A})] - \Pr\left[\mathbf{Expt}_{g}(\mathcal{A})\right]$.

If we use $\mathcal{F}$ in a protocol that requires that the security can be broken with at most negligible probability for a security parameter η, we also want $\mathbf{Adv}_{\mathcal{F}}(\mathcal{A})$ to be negligible in η. Therefore, we use the following security definition for pseudorandom functions:

Definition 6.4.2. *$\mathcal{F}$ is a secure pseudorandom function family if, for all probabilistic polynomial time algorithms $\mathcal{A}$, the adversarial advantage $\mathbf{Adv}_{\mathcal{F}}(\mathcal{A})$ in the security game defined in Definition 6.4.1 is bounded by a negligible quantity in the security parameter η.*

6.4.2 Anonymity Analysis

Theorem 6.4.1 (Sender Anonymity for OrgAn)**.** *Assuming $\mathcal{F}()$ is a computationally secure pseudorandom function, the OrgAn provides sender anonymity as defined in Definition 2.1.2 with negligible δ against any global passive adversary $\mathcal{A}$, as long as at least two users and one setup server are honest.*

Proof. Without loss of generality let us assume that users u_1 and u_2 are honest and the message associated sent by them in a given round are x_1 and x_2 respectively. Let us also assume that only one setup server G_1 is honest. Now we prove security in two part parts:

1. First we use a modified version $OrgAn^*$ of the protocol OrgAn and show that the adversary has a negligible advantage against $OrgAn^*$. In $OrgAn^*$, the user u_1 uses a random function $\mathcal{F}_{rand}(\cdot)$ instead of $\mathcal{F}(\mathbf{r}_1, \cdot)$ as the masks to compute the ciphertexts; and the user u_2 uses $\mathcal{F}(\mathbf{r}_1 + \mathbf{r}_2, \cdot) - \mathcal{F}_{rand}(\cdot)$.

2. Next we show that, if an adversary $\mathcal{A}_{anon}$ wins the game against OrgAn in the anonymity game, we can construct an adversary $\mathcal{A}_{PRF}$ that can win the PRF game.

As our first step, we consider the anonymity game with the protocol $OrgAn^*$. In $OrgAn^*$, all the protocol parties except u_1, u_2 and G_1 behave exactly the same as OrgAn. However, in the hypothetical protocol $OrgAn^*$ we assume that u_1 and u_2 collude in the following way: for a given slot t the user u_1 uses $\mathcal{F}_{rand}(t)$ as the mask to compute ciphertext $c_1(t) = x_1^t + \mathcal{F}_{rand}(t)$, and the user u_2 compute ciphertext $c_2(t) = x_2^t + \mathcal{F}(\mathbf{r}_1 + \mathbf{r}_2, t) - \mathcal{F}_{rand}(t)$. For the time being, let us consider only one round and we will extend the argument for multiple rounds shortly.

In this hypothetical protocol $OrgAn^*$, we can assume that the users u_1 and u_2 can exchange information about $\mathbf{r}_1, \mathbf{r}_2$ and $\mathcal{F}_{rand}()$ with each other.

Claim 12. *The protocol $OrgAn^*$ provides sender anonymity with $\delta = 0$ against any global passive adversary $\mathcal{A}$, for a one-round protocol run.*

Proof of Claim S. ince $\mathcal{F}_{rand}()$ is a random function, the value $\mathcal{F}_{rand}(t)$ can be thought of as being chosen at random. Let, $f_1 = \mathcal{F}_{rand}(1)$ and $f_2 = \mathcal{F}(\mathbf{r}_1 + \mathbf{r}_2, 1) - \mathcal{F}_{rand}(1)$. Then the adversary $\mathcal{A}$ has the following set of equations for slot 1 with x_1, x_2, f_1, f_2 as unknowns (we skip the group notations for simplicity),

1. $x_1 + x_2 = a_1$

2. $x_1 + f_1 = a_2$

3. $f_2 + x_2 = a_3$

4. $f_1 + f_2 = a_4$

and the adversary knows $\langle x_1, x_2 \rangle = \langle b_1, b_2 \rangle$ or $\langle b_2, b_1 \rangle$, for some observed values of a_1, a_2, a_3, a_4, b_1, b_2. Note that the above equation system has a rank of at most 3; both $\langle b_1, b_2 \rangle$ or $\langle b_2, b_1 \rangle$

will yield valid values of $f1$ and $f2$. Therefore, slot 1 does not reveal anything about who sent x_1 or x_2.

Since $\mathcal{F}_{rand}$ is a random function, $\mathcal{F}_{rand}(t)$ is unrelated from $\mathcal{F}_{rand}(t)$ for any $t \neq t$. And hence, a similar argument can be extended for any other slot t as well, independent of slot t. Since the overall equation system to retrieve all the messages in a round is an identity, the adversary has $\delta = 0$ advantage in the sender anonymity game against the protocol $OrgAn^*$ for a one-round protocol run. ◇

Now let us consider the scenario when the protocol $OrgAn^*$ is run for many rounds. For every round d and slot t, the user u_1 can use $(d\mathsf{N} + t)$ as the input to the random function $\mathcal{F}_{rand}()$. In that case, the input to $\mathcal{F}_{rand}()$ is never repeated, and $OrgAn^*$ provides sender anonymity with $\delta = 0$ even for a multi-round protocol run. We skip the formal claim statement and proof here, since they are similar to that of Claim 12.

Now that we have proved δ-sender anonymity for $OrgAn^*$ with $\delta = 0$, we proceed to the next step where we prove the anonymity of OrgAn. We show that, if there exist an adversary $\mathcal{A}_{anon}$ that breaks sender anonymity for protocol $OrgAn$, we can construct an adversary $\mathcal{A}_{PRF}$ that break the security assumption on pseudorandom function $\mathcal{F}$.

Claim 13. *If there exist a PPT adversary $\mathcal{A}_{anon}$ with an adversarial advantage δ against the protocol $OrgAn$ in the sender anonymity game defined in Definition 2.1.2, there exist an adversary $\mathcal{A}_{PRF}$ that can distinguish between $\mathcal{F}$ and $\mathcal{F}_{rand}$ with probability at least δ in the PRF game defined in Definition 6.4.1.*

Proof of Claim W. e start with the construction of $\mathcal{A}_{PRF}$: our adversary $\mathcal{A}_{PRF}$ of the PRF game will run the whole sender anonymity game as the challenger, except one setup server G_1 (as per our threat model at least one setup server is honest, and without loss of generality we assume that to be G_1).

The key $\mathbf{K}$ for the PRF game is decided based on the random number $\mathbf{r}_{11}$ picked by G_1. We pick, $\mathbf{K} = \sum_j \mathbf{r}_{1j} = \mathbf{r}_1$ such that $\mathbf{r}_1 = \mathbf{K}$. Since G_1 as an independent honest party that does not collude with $\mathcal{A}_{PRF}$ or $\mathcal{A}_{anon}$[1], $\mathcal{A}_{PRF}$ does not know $\mathbf{r}_{11}$ or $\mathbf{K}$.

[1]↑More formally G_1 can modeled similar to hybrid functionalities in UC framework, and then the security game can be defined in that hybrid functionality setting. We skip the rigorous formalization here.

$\mathcal{A}_{PRF}$ runs each round of the sender anonymity game in the following way: for each slot value t $\mathcal{A}_{PRF}$ queries the PRF game with input value t and receives a value f_t. $\mathcal{A}_{PRF}$ asks the user u_1 in the sender anonymity game to use f_t to compute $c_1(t) = x_i^t + f_t$. Similarly, $\mathcal{A}_{PRF}$ asks u_2 to use $c_2(t) = x_i^t + \mathcal{F}(\mathbf{r}_1 + \mathbf{r}_2, t) - f_t$. $\mathcal{A}_{PRF}$ runs the sender anonymity game until $\mathcal{A}_{anon}$ halts. $\mathcal{A}_{PRF}$ returns 1 if and only if $\mathcal{A}_{anon}$ wins the sender anonymity game. When f_t is an output of $\mathcal{F}_{rand}()$ the adversary $\mathcal{A}_{PRF}$ is effectively running $OrgAn^*$, however, when $f_t = \mathcal{F}(\mathbf{K}, t)$ it is running OrgAn. If $\mathcal{A}_{anon}$ has an advantage of δ in the sender anonymity game against OrgAn, there would be a difference of at least δ in the probability $\mathcal{A}_{PRF}$ outputs 1 when f_t is the output of $\mathcal{F}_{rand}()$ vs when it is the output of $\mathcal{F}()$. $\diamond$

Following the above claim, if the adversarial advantage of $\mathcal{A}_{anon}$ is non-negligible against OrgAn, so is the adversarial advantage of $\mathcal{A}_{PRF}$ in the PRF game — which contradicts the assumption that $\mathcal{F}$ is a secure pseudorandom function. $\square$

The above theorem shows that the protocol OrgAn provides sender anonymity with negligible adversarial advantage against a global passive adversary.

7. SUMMARY

This dissertation can act as a guideline towards building secure AC protocols, since it provides the fundamental necessary constraints required to achieve anonymity. Further, it presents two provably secure protocol constructions inspired by the proven guidelines.

Fundamental Constraints For Anonymous Communication Protocols:

This work derives necessary constraints for sender anonymity and recipient anonymity, and thereby presents necessary constraints that are crucial for understanding bi-directional anonymous communication: sender anonymity for hiding the sender and recipient anonymity for hiding the recipient of a message. When analyzing our results we notice that there are three distinct relevant invariants upon which the possibility for anonymity depends. These are: the protocol respects its limitations, the user distribution allows for anonymity and in case of partially compromised AC protocols, a user's message needs to travel through at least one honest protocol party.

We further identify a strong class of AC protocols that utilize what we call user coordination, an ability by which users work together to introduce uncertainty. Even this additional power does not fundamentally change the requirements on latency overhead ℓ and bandwidth overhead B – except that excessive bandwidth on its own is sufficient to provide strong anonymity, independent of latency or even the level of compromisation. In the absence of this extreme case, a combination of latency and bandwidth overhead is still necessary. In addition to confirming this crucial insight, our formal analysis yields additional requirements for strong anonymity based on the number of compromised nodes c: if $\mathsf{c} > 0$ then the latency overhead must grow ($\ell \notin O(1)$); if $\mathsf{c}/\mathsf{K} \geq 1/2$ and $\ell \leq \log(\eta)$, then more and more messages must be in the system, i.e., ℓp cannot be constrained by any constant; if $\mathsf{K} - \mathsf{c} \in O(1)$, such as in the Anytrust assumption, then either $\ell > \eta^2$ or $\ell p > \sqrt[4]{\eta}$ are required.

Future work on ACNs can directly build on our insights; our formulas indicate that user coordination in the style of DC-nets (or Herbivore or Dissent-AT) can reduce the gap to the universal necessary constraint ($\ell p \geq 1 - \epsilon(\eta)$) significantly. For closing the gap, however, our results point to ACN designs outside of our wide class of ACNs (see Section 4.1.1 for

a thorough discussion). Protocol designers thus face a choice: settle for a weaker notion of anonymity, come up with novel techniques, sacrifice reliability or adhere to the latency and bandwidth overheads described in this work.

Provably Secure Constructions:

This work presents two provably secure constructions of AC protocols: (1) a mixnet-based protocol *Streams* that achieves strong pairwise unlinkability, (2) a DC-net inspired protocol *OrgAn* that achieves strong sender anonymity.

Streams. Streams is a novel mixnet-type protocol that realizes similar security properties as a trusted third party stop-and-go mix while allowing a fraction of mixnodes to be compromised by the adversary, with a latency overhead of several seconds (depending on the proportion of compromised nodes). As long as each message stays in the system for at least the given amount of time before being delivered, our protocol provides the same security guarantees comparable to that of a trusted third party.

With our supernode construction, the latency required to maintain the same level of anonymity does not grow with the number of users. That shows Streams with supernodes is much better suited to scale with number of users. Other protocol like Loopix [1], Hornet [14], Tor [49] can employ this supernode structures with minor modifications towards improving their anonymity with better scalability.

OrgAn. OrgAn is a new DC-net inspired AC protocol using key-homomorphic pseudorandom functions (PRF) and Newton's power sums in the client/relay/server model. Similar to other DC-net based protocols, it provides strong sender anonymity guarantees with resistance against intersection attacks. Unlike existing DC-net based protocols, it avoids the overhead of slot and key agreement per round, and thus, presents an efficient way of achieving user coordination.

An AC protocol that combines the techniques of mixnets and user coordination to provide a tunable trade-off between latency and bandwidth overhead still remains an interesting future work.

BIBLIOGRAPHY

[1] A. Piotrowska, J. Hayes, T. Elahi, S. Meiser, and G. Danezis, "The loopix anonymity system," in *Proc. 26th USENIX Security Symposium*, 2017.

[2] J. van den Hooff, D. Lazar, M. Zaharia, and N. Zeldovich, "Vuvuzela: Scalable private messaging resistant to traffic analysis," in *Proc. 25th ACM Symposium on Operating Systems Principles (SOSP 2015)*, 2015.

[3] D. Lazar and N. Zeldovich, "Alpenhorn: Bootstrapping secure communication without leaking metadata," 10 2016.

[4] D. Chaum, "Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms," *Communications of the ACM*, vol. 4, no. 2, pp. 84–88, 1981.

[5] D. I. Wolinsky, H. Corrigan-Gibbs, B. Ford, and A. Johnson, "Dissent in Numbers: Making Strong Anonymity Scale," in *10th USENIX Symposium on Operating Systems Design and Implementation (OSDI 12)*, 2012, pp. 179–182.

[6] H. Corrigan-Gibbs, D. Boneh, and D. Mazières, "Riposte: An anonymous messaging system handling millions of users," in *2015 IEEE Symposium on Security and Privacy*. IEEE, 2015, pp. 321–338.

[7] S. Le Blond, D. Choffnes, W. Caldwell, P. Druschel, and N. Merritt, "Herd: A Scalable, Traffic Analysis Resistant Anonymity Network for VoIP Systems," in *Proc. ACM Conference on Special Interest Group on Data Communication (SIGCOMM 2015)*, 2015, pp. 639–652.

[8] A. Kwon, D. Lazar, S. Devadas, and B. Ford, "Riffle: An Efficient Communication System With Strong Anonymity," in *Proc. Privacy Enhancing Technologies Symposium (PETS 2016)*, 2016, pp. 115–134.

[9] S. Angel and S. Setty, "Unobservable Communication over Fully Untrusted Infrastructure," in *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation (OSDI)*. USENIX Association, 2016, pp. 551–569.

[10] S. Goel, M. Robson, M. Polte, and E. Sirer, "Herbivore: A scalable and efficient protocol for anonymous communication," 2003, https://www.cs.cornell.edu/people/egs/herbivore/herbivore.pdf.

[11] P. Golle and A. Juels, "Dining cryptographers revisited," in *Proc. of Eurocrypt 2004*, 2004.

[12] D. Chaum, "The dining cryptographers problem: Unconditional sender and recipient untraceability," *Journal of Cryptology*, vol. 1, no. 1, pp. 65–75, 1988.

[13] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The Second-Generation Onion Router," in *Proc. 13th USENIX Security Symposium (USENIX)*, 2004, pp. 303–320.

[14] C. Chen, D. E. Asoni, D. Barrera, G. Danezis, and A. Perrig, "HORNET: High-speed onion routing at the network layer," in *Proc. ACM Conference on Computer and Communications Security (CCS)*, 2015, pp. 1441–1454.

[15] T. Ruffing, P. Moreno-Sanchez, and A. Kate, "P2P Mixing and Unlinkable Bitcoin Transactions," in *Proc. 25th Annual Network & Distributed System Security Symposium (NDSS)*, 2017.

[16] "The Tor Project," https://www.torproject.org/, accessed in Nov 2018.

[17] A. Johnson, C. Wacek, R. Jansen, M. Sherr, and P. Syverson, "Users get routed: Traffic correlation on tor by realistic adversaries," in *Proc. ACM SIGSAC conference on Computer & communications security 2013*, 2013, pp. 337–348.

[18] L. Øverlier and P. F. Syverson, "Locating Hidden Servers," in *Proc. 27th IEEE Symposium on Security and Privacy*, 2006, pp. 100–114.

[19] S. J. Murdoch and G. Danezis, "Low-cost traffic analysis of Tor," in *Proc. IEEE Symposium on Security and Privacy 2005*, 2005.

[20] K. S. Bauer, D. McCoy, D. Grunwald, T. Kohno, and D. C. Sicker, "Low-resource routing attacks against tor," 2007, pp. 11–20.

[21] Y. Sun, A. Edmundson, L. Vanbever, O. Li, J. Rexford, M. Chiang, and P. Mittal, "RAPTOR: Routing attacks on privacy in Tor," in *Proc. 24th USENIX Security Symposium*, 2015.

[22] R. Jansen, F. Tschorsch, A. Johnson, and B. Scheuermann, "The sniper attack: Anonymously deanonymizing and disabling the Tor network," in *Proc. Network and Distributed Security Symposium - NDSS '14*, 2014.

[23] Y. Gilad and A. Herzberg, "Spying in the Dark: TCP and Tor Traffic Analysis," in *Proc. 12th Privacy Enhancing Technologies Symposium (PETS 2012)*, 2012.

[24] D. Das, S. Meiser, E. Mohammadi, and A. Kate, "Anonymity trilemma: Strong anonymity, low bandwidth overhead, low latency - choose two," in *2018 IEEE Symposium on Security and Privacy (SP)*, May 2018, pp. 108–126, extended version under https://eprint.iacr.org/2017/954.

[25] D. Das, S. Meiser, E. Mohammadi, and A. Kate, "Comprehensive anonymity trilemma: User coordination is not enough," *Proceedings on Privacy Enhancing Technologies*, vol. 2020, pp. 356–383, 07 2020.

[26] The Tor Blog, "One cell is enough to break Tor's anonymity," https://blog.torproject.org/one-cell-enough-break-tors-anonymity, accessed Nov 2020.

[27] S. Chakravarty, M. V. Barbera, G. Portokalidis, M. Polychronakis, and A. D. Keromytis, "On the effectiveness of traffic analysis against anonymity networks using flow records," in *Proc. 15th International Conference on Passive and Active Measurement*, 2014, pp. 247–257.

[28] N. Gelernter and A. Herzberg, "On the limits of provable anonymity," in *Proc. Workshop on Privacy in the Electronic Society (WPES 2013)*, 2013, pp. 225–236.

[29] A. Hevia and D. Micciancio, "An indistinguishability-based characterization of anonymous channels," in *Proc. Eighth International Symposium on Privacy Enhancing Technologies (PETS 2008)*, N. Borisov and I. Goldberg, Eds., 2008, pp. 24–43.

[30] A. Serjantov, R. Dingledine, and P. Syverson, "From a trickle to a flood: Active attacks on several mix types," in *5th Information Hiding Workshop (IH 2002)*, 2003, pp. 36–52.

[31] D. Lazar, Y. Gilad, and N. Zeldovich, "Karaoke: Distributed private messaging immune to passive traffic analysis," in *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*. Carlsbad, CA: USENIX Association, Oct. 2018, pp. 711–725. [Online]. Available: https://www.usenix.org/conference/osdi18/presentation/lazar

[32] A. Kwon, H. Corrigan-Gibbs, S. Devadas, and B. Ford, "Atom: Horizontally scaling strong anonymity," in *Proceedings of the 26th Symposium on Operating Systems Principles*, ser. SOSP '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 406–422. [Online]. Available: https://doi.org/10.1145/3132747.3132755

[33] N. Tyagi, Y. Gilad, D. Leung, M. Zaharia, and N. Zeldovich, "Stadium: A distributed metadata-private messaging system," 10 2017, pp. 423–440.

[34] D. Kesdogan, J. Egner, and R. Büschkes, "Stop-and-go MIXes: Providing probabilistic anonymity in an open system," in *Proc. Information Hiding Workshop (IH 1998)*, 1998.

[35] H. Corrigan-Gibbs and B. Ford, "Dissent: Accountable Anonymous Group Messaging," in *CCS10*, 2010, pp. 340–350.

[36] L. Barman, I. Dacosta, M. Zamani, E. Zhai, B. Ford, J. Hubaux, and J. Feigenbaum, "Prifi: A low-latency local-area anonymous communication network," *CoRR*, vol. abs/1710.10237, 2017. [Online]. Available: http://arxiv.org/abs/1710.10237

[37] H. Corrigan-Gibbs, D. I. Wolinsky, and B. Ford, "Proactively Accountable Anonymous Messaging in Verdict," in *Proc. 22nd USENIX Security Symposium*, 2013, pp. 147–162.

[38] S. Le Blond, D. Choffnes, W. Zhou, P. Druschel, H. Ballani, and P. Francis, "Towards Efficient Traffic-analysis Resistant Anonymity Networks," in *Proc. ACM SIGCOMM 2013*, 2013, pp. 303–314.

[39] M. Backes, A. Kate, P. Manoharan, S. Meiser, and E. Mohammadi, "AnoA: A Framework For Analyzing Anonymous Communication Protocols," *Journal of Privacy and Confidentiality (JPC)*, vol. 7(2), no. 5, 2016.

[40] K. Jensen, *Colored Petri Nets. Basic Concepts, Analysis Methods and Practical Use.*, 1997, vol. 3.

[41] W. Reisig, *Primer in Petri Net Design*, 1st ed., 1992.

[42] T. K. Srikanth and S. Toueg, "Simulating authenticated broadcasts to derive simple fault-tolerant algorithms," *Distributed Computing*, vol. 2, no. 2, pp. 80–94, 1987.

[43] R. Gennaro, M. O. Rabin, and T. Rabin, "Simplified VSS and fact-track multiparty computations with applications to threshold cryptography," in *Proc. ACM PODC*, 1998, pp. 101–111.

[44] M. Backes, P. Manoharan, and E. Mohammadi, "TUC: Time-sensitive and Modular Analysis of Anonymous Communication," IACR ePrint Archive Report 2013/664, 2013, http://www.infsec.cs.uni-saarland.de/~mohammadi/paper/tuc.pdf.

[45] D. Dolev, R. Reischuk, and H. R. Strong, "Early stopping in byzantine agreement," *J. ACM*, vol. 37, no. 4, pp. 720–741, 1990.

[46] L. M. Kristensen, S. Christensen, and K. Jensen, "The practitioner's guide to coloured petri nets," *International Journal on Software Tools for Technology Transfer (STTT)*, vol. 2, no. 2, pp. 98–132, 1998.

[47] G. Danezis, C. Diaz, C. Troncoso, and B. Laurie, "Drac: An architecture for anonymous low-volume communications," in *Proc. 10th Privacy Enhancing Technologies Symposium (PETS 2010)*, 2010.

[48] P. Mittal, M. Wright, and N. Borisov, "Pisces: Anonymous communication using social networks," in *ndss13*, 2013.

[49] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The second-generation onion router," in *Proceedings of the 13th Conference on USENIX Security Symposium - Volume 13*, ser. SSYM'04. USA: USENIX Association, 2004, p. 21.

[50] G. R. Blakley and C. Meadows, "Security of ramp schemes," in *Advances in Cryptology*, 1985, pp. 242–268.

[51] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(leveled) fully homomorphic encryption without bootstrapping," in *Proceedings of the 3rd Innovations in Theoretical Computer Science (ITCS) Conference*, 2012, pp. 309–325.

[52] Anonymity Trilemma Project Webpage, "Anonymity trilemma: Strong anonymity, low bandwidth overhead, low latency overhead—choose two," https://freedom.cs.purdue.edu/projects/anonymity/trilemma/.

[53] N. Alexopoulos, A. Kiayias, R. Talviste, and T. Zacharias, "MCMix: Anonymous Messaging via Secure Multiparty Computation," in *Proceedings of the 26th USENIX Security Symposium*. USENIX Association, 2017, pp. 1217–1234.

[54] D. M. Goldschlag, M. G. Reed, and P. F. Syverson, "Onion Routing," *Commun. ACM*, vol. 42, no. 2, pp. 39–41, 1999.

[55] N. S. Evans, R. Dingledine, and C. Grothoff, "A Practical Congestion Attack on Tor Using Long Paths," 2009, pp. 33–50.

[56] D. Lazar, Y. Gilad, and N. Zeldovich, "Yodel: Strong metadata security for voice calls," in *Proceedings of the 27th ACM Symposium on Operating Systems Principles*, ser. SOSP '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 211–224. [Online]. Available: https://doi.org/10.1145/3341301.3359648

[57] S. Eskandarian, H. Corrigan-Gibbs, M. Zaharia, and D. Boneh, "Express: Lowering the cost of metadata-hiding communication with cryptographic privacy," *ArXiv*, vol. abs/1911.09215, 2019.

[58] D. Lu, T. Yurek, S. Kulshreshtha, R. Govind, A. Kate, and A. Miller, "Honeybadgermpc and asynchromix: Practical asynchronous mpc and its application to anonymous communication," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 887–903.

[59] I. Abraham, B. Pinkas, and A. Yanai, "Blinder – scalable, robust anonymous committed broadcast," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '20, 2020, p. 1233–1252.

[60] C. Kuhn, M. Beck, and T. Strufe, "Breaking and (Partially) Fixing Provably Secure Onion Routing," in *Proceedings of the 41st IEEE Symposium on Security and Privacy*. IEEE, 2020, pp. 168–185.

[61] A. Kate, G. M. Zaverucha, and I. Goldberg, "Pairing-based onion routing with improved forward secrecy," *ACM Transactions on Information and System Security (TISSEC)*, vol. 13, no. 4, pp. 1–32, 2010.

[62] G. Danezis and I. Goldberg, "Sphinx: A Compact and Provably Secure Mix Format," 2009, pp. 269–282.

[63] I. T. L. Computer Security Division, "Interoperable randomness beacons: Csrc." [Online]. Available: https://csrc.nist.gov/projects/interoperable-randomness-beacons

[64] Drand, "Drand - a distributed randomness beacon daemon." [Online]. Available: https://github.com/drand/drand

[65] M. O. Rabin, "Randomized byzantine generals," in *24th Annual Symposium on Foundations of Computer Science (SFCS 1983)*. IEEE, 1983, pp. 403–409.

[66] A. Bhat, N. Shrestha, A. Kate, and K. Nayak, "Randpiper - reconfiguration-friendly random beacons with quadratic communication," *IACR Cryptol. ePrint Arch.*, no. 1590, 2020. [Online]. Available: https://eprint.iacr.org/2020/1590

[67] E. Syta, P. Jovanovic, E. K. Kogias, N. Gailly, L. Gasser, I. Khoffi, M. J. Fischer, and B. Ford, "Scalable bias-resistant distributed randomness," in *2017 IEEE Symposium on Security and Privacy (SP)*. Ieee, 2017, pp. 444–460.

[68] P. Schindler, A. Judmayer, N. Stifter, and E. Weippl, "Hydrand: Practical continuous distributed randomness," in *2020 IEEE Symposium on Security and Privacy (SP). IEEE*, 2020.

[69] C. Cachin, K. Kursawe, and V. Shoup, "Random oracles in constantinople: Practical asynchronous byzantine agreement using cryptography," *Journal of Cryptology*, vol. 18, no. 3, pp. 219–246, 2005.

[70] T. Hanke, M. Movahedi, and D. Williams, "Dfinity technology overview series, consensus system," *arXiv preprint arXiv:1805.04548*, 2018.

[71] M. Haahr, "True random number service." [Online]. Available: https://www.random.org/

[72] "blockchain oracle service, enabling data-rich smart contracts." [Online]. Available: https://provable.xyz/

[73] "Generate random numbers for smart contracts using chainlink vrf." [Online]. Available: https://docs.chain.link/docs/chainlink-vrf

[74] D. Chaum, D. Das, F. Javani, A. Kate, A. Krasnova, J. de Ruiter, and A. T. Sherman, "cmix: Mixing with minimal real-time asymmetric cryptographic operations," in *15th International Conference on Applied Cryptography and Network Security 2017*, 2017.

[75] R. Canetti, "Universally Composable Security: A New Paradigm for Cryptographic Protocols," in *FOCS01*, 2001, pp. 136–145.

[76] S. Gajek, M. Manulis, O. Pereira, A.-R. Sadeghi, and J. Schwenk, "Universally composable security analysis of tls," in *Provable Security*, J. Baek, F. Bao, K. Chen, and X. Lai, Eds. Springer Berlin Heidelberg, 2008, pp. 313–327.

[77] R. 8446, "The transport layer security (tls) protocol version 1.3," https://tools.ietf.org/html/rfc8446, accessed April 2021.

[78] A. Kate and I. Goldberg, "Using Sphinx to Improve Onion Routing Circuit Construction," in *FC10*, 2010, pp. 359–366.

[79] Deepmind, "Anonymous messaging using mix networks," https://github.com/deepmind/loopix-messaging, accessed in January 2021.

[80] M. Eberl, "Fisher-yates shuffle," *Arch. Formal Proofs*, vol. 2016, 2016.

[81] M. Ando, A. Lysyanskaya, and E. Upfal, "On the complexity of anonymous communication through public networks," *CoRR*, vol. abs/1902.06306, 2019. [Online]. Available: http://arxiv.org/abs/1902.06306

[82] R. Henry, A. Herzberg, and A. Kate, "Blockchain access privacy: Challenges and directions," *IEEE Security Privacy*, vol. 16, no. 4, pp. 38–45, 2018.

[83] D. Das, S. Meiser, E. Mohammadi, and A. Kate, "Comprehensive anonymity trilemma: User coordination is not enough," *Proc. Priv. Enhancing Technol.*, vol. 2020, no. 3, pp. 356–383, 2020.

[84] H. W. Gould, "The girard-waring power sum formulas for symmetric functions, and fibonacci sequences," *Fibonacci Quarterly*, vol. 37, no. 2, pp. 135–140, 1999, https://www.fq.math.ca/Issues/37-2.pdf.

[85] D. Boneh, K. Lewi, H. Montgomery, and A. Raghunathan, "Key homomorphic prfs and their applications," in *Advances in Cryptology – CRYPTO 2013*, R. Canetti and J. A. Garay, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 410–428.

[86] C. P. Schnorr, "Efficient identification and signatures for smart cards," in *Advances in Cryptology — CRYPTO' 89 Proceedings*, G. Brassard, Ed. New York, NY: Springer New York, 1990, pp. 239–252.

[87] D. Pointcheval and J. Stern, "Security arguments for digital signatures and blind signatures," *Journal of Cryptology*, vol. 13, 10 2001.

[88] RFC 8446, "The Transport Layer Security (TLS) Protocol Version 1.3," https://tools.ietf.org/html/rfc8446, accessed April 2021.

[89] A. Banerjee, C. Peikert, and A. Rosen, "Pseudorandom functions and lattices," in *Proceedings of the 31st Annual International Conference on Theory and Applications of Cryptographic Techniques*, ser. EUROCRYPT'12. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 719–737. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-29011-4_42

A. ADDITIONAL CONTENTS FOR INTERESTED READERS

A.1 Interesting calculations Related to Impossibility Results

A.1.1 Calculating the probability of a specific user sending a message in a span of d rounds, for unsynchronized user message distribution

Here we derive an upper bound on the probability that a specific user (Bob) sends a message in a given span of d rounds, given that the protocol knows the frequency distribution of the messages over rounds. This approximates the cases where protocols can choose the delay of a message depending on the density of messages at different times.

Consider the following indicator random variables: $X^{(1)}, X^{(2)}, \ldots, X^{(\mathsf{N})}$, each $X^{(i)}$ denoting if user i sends a message or not in a span of d rounds. Since every user acts independent of all other users, $X^{(i)}$s are mutually independent, and $X^{(i)}$ can be defined as,

$$X^{(i)} = \begin{cases} 0 & \text{with probability } (1-p)^d \\ 1 & \text{with probability } (1-(1-p)^d). \end{cases}$$

We further denote $X = \sum_{i=1}^{\mathsf{N}} X^{(i)}$. The expectation of X can be calculated as, $\mathbb{E}[X] = \sum_{i=1}^{\mathsf{N}} \mathbb{E}[X^{(i)}] = \mathsf{N}(1-(1-p)^d) = \mu$.

Using Markov's Inequality we can say, $\Pr[X \geq 2\mu] \leq \frac{1}{2}$.

At least one message is sent by our chosen user Bob is denoted by the event Y. If the total number of messages in the span of d rounds is $x \in \{0, \ldots, \mathsf{N}\}$, $\Pr[Y|X=x] \leq \frac{x}{\mathsf{N}}$.

For $2\mu \leq \mathsf{N}$, we can derive,

$$\begin{aligned} & f_p^{SA}(d) = \Pr[Y] \\ =& \Pr[X \geq 2\mu] \cdot \Pr[Y|X \geq 2\mu] + \Pr[X < 2\mu] \cdot \Pr[Y|X < 2\mu] \\ \leq& \Pr[X \geq 2\mu] \cdot \Pr[Y|X = \mathsf{N}] + \Pr[X < 2\mu] \cdot \Pr[Y|X = 2\mu] \\ \leq& \frac{1}{2} \cdot \frac{\mathsf{N}}{\mathsf{N}} + \left(1 - \frac{1}{2}\right) \cdot \frac{2\mu}{\mathsf{N}} = \frac{1}{2} + \left(1 - (1-p)^d\right). \end{aligned}$$

If $2\mu > \mathsf{N}$, we get $f(d) = \Pr[Y] = 1$. Using Chernoff bound, we can derive derive a tighter bound $\Pr[X \geq 2\mu] = \sigma(d) \leq \exp\left(-2(\mu^2/\mathsf{N}^2)\mathsf{N}\right)$. However, since we are interested in impossibility results, and the difference is a constant factor $\frac{1}{2}$, we utilize the result obtained

by using Markov's inequality. We formally define the probability of Bob sending a message in the given d rounds as, $f_p^{SA}(d) = min\left(1, \frac{1}{2} + \left(1 - (1-p)^d\right)\right)$.

However, when we analyze the possibility of strong anonymity, if $pd \leq 1$ we use $f_p^{SA}(d) = \left(1 - (1-p)^d\right)$ instead. Because $\sigma(d)$ becomes negligible in N, thus negligible in η.

Lemma 3. *When* $pd \leq 1$, $\sigma(d) \leq \exp\left(-2(\mu^2/\mathsf{N}^2)\mathsf{N}\right)$ *is negligible in* N *for* $\mu = \mathsf{N}(1-(1-p)^d)$.

Proof. For $pd \leq 1$ we can say,

$$\begin{aligned} pd \leq 1 &\Rightarrow (1-p)^d < \frac{1}{e} \\ &\Rightarrow 1-(1-p)^d > \frac{1}{2} \\ &\Rightarrow \frac{\mu^2}{\mathsf{N}^2} > \frac{1}{4} \end{aligned}$$

Therefore, $\sigma(d) \leq \exp\left(-2(\mu^2/\mathsf{N}^2)\mathsf{N}\right) \leq \exp\left(-\mathsf{N}/2\right)$ — always negligible in N. □

A.1.2 Analyze average case of the user distribution, to derive impossibility conditions for strong/quadratic anonymity

Lemma 4. *Let* R *be the set of all possible* sequences of execution *of an AC protocol. Let* $Runs \in R$ *be a random variable denoting the sequence of execution. Suppose, for a set of sequences of execution* $R \subset R$, $\Pr[Runs \in R]$ *is* μ*, and* μ *non-negligible. If the protocol can not provide strong anonymity for any execution* $o \in R$*, the protocol can not provide strong anonymity overall.*

Proof. Suppose Y denotes the event that the adversary wins, and o^* is the element in R for which the probability that the adversary wins is maximum, i.e., $\Pr[Y \mid o^*] \geq \Pr[Y \mid o]$ for all $o \in R$. Suppose, $\Pr[Y \mid o^*] = \nu$. Then we can say,

$$\begin{aligned}
\Pr[Y] &= \sum_{o \in R} \Pr[Y \mid Runs = o] \cdot \Pr[Runs = o] \\
&= \sum_{o \in R} \Pr[Y \mid Runs = o] \cdot \Pr[Runs = o] \\
&\quad + \sum_{o \in R \setminus R} \Pr[Y \mid Runs = o] \cdot \Pr[Runs = o] \\
&\leq \sum_{o \in R} \Pr[Y \wedge Runs = o^*] \cdot \Pr[Runs = o] + S_R \\
&= \Pr[Y \wedge Runs = o^*] \cdot \Pr[Runs \in R] + S_R \\
&= \nu \cdot \mu + S_R
\end{aligned}$$

where $S_R = \sum_{o \in R \setminus R} \Pr[Y \mid Runs = o] \cdot \Pr[Runs = o]$. We know μ and ν both are non-negligible. Therefore, $\Pr[Y]$ is non-negligible. □

The above lemma provides us with a very helpful insight for unsynchronized user message distribution. Suppose X denotes the total number of messages in a given slice of $\hat{\ell}$ rounds, and R denotes the set of all sequences of execution where $X < \mathbb{E}[X]$. For two values x_1 and x_2 drawn from X and $x_1 > x_2$, the protocol has a better chance for anonymity with $X = x_1$. Therefore, if we are analyzing the possibility of strong anonymity, it is enough to analyze the average case scenario in most of the cases (e.g., $p\hat{\ell} \in O(1)$ – in which case we can replace $f_p^{SA}(d)$ with $\left(1 - (1-p)^d\right)$ for a given slice of d rounds). Additionally, we can use $\delta \geq \left(1 - \frac{B_{\text{eff}}}{\mathsf{N}-1}\right)\left[1 - g(Z) \times f_p^{SA}(\hat{\ell})\right]$, where $Z = min\left(\hat{\ell}, 2B+1\right)$ to analyze strong anonymity. Because, by Markov inequality, the number of additional shares for a message will be bounded by $2B$ with probability at least $\frac{1}{2}$.

A.2 Expressing Protocols in the Petri net model

A.2.1 Modeling DC net

Here we show how to model an actual DC net type protocol using our Petri net model when the users use user coordination. Specifically we pick up the *short DC net* protocol proposed by *Golle and Juels* [11], and present M_{DC} which models the aforementioned protocol.

We model a DC net protocol with N participants, where $\mathcal{S} = \mathsf{P}$, $|\mathcal{S}| = |\mathsf{P}| = \mathsf{N}$. We denote the parties with $P_1, \ldots, P_N$. The protocol can be denoted by Π_{DC} ={paramgen, keydist, post, verify, extract}[1] - as described below.

- *paramgen*: In Π_{DC}, *paramgen* is executed by a trusted entity and the output is published. Since we are mainly interested in the anonymity game, we consider that *paramgen* step is executed by our honest challenger and happens outside the protocol run, and the output is globally known (to all the transitions T_{P_i}).
- *keydist*: using the output of *paramgen*, this step yields for each party P_i a private key x_i and a corresponding public key y_i. In $prot_{DC}$, the above key generation part is done by a trusted entity, and hence we consider that it is done by our honest challenger and for each party P_i the private-public keypair x_i, y_i is already known to the corresponding transition function T_{P_i}. As part of protocol each party P_i publishes its public key y_i. Additionally, each party P_j receives from P_i a share of private key $x_{i,j}$ and a share of public key $y_{i,j}$, where the keys are shared in a (k, N) threshold manner for a parameter $k \leq \mathsf{N}$.
- *post*: Each player P_i generates a vector of random pads $W_i = \{W_i(1), W_i(2), \ldots, W_i(N)\}$[2] using x_i. Π_{DC} does not handle *collisions*, instead assumes that the players decide their positions by a consensus protocol. Similarly our model assumes that each party P_i knows its position, and assume the position is q_i (but not known to the adversary). Then each player P_i computes the vector V_i such that $V_i(w) = W_i(w)$ for all $w \neq i$ and $V_i(w) = W_i(w) \oplus m_i$ for $w = q_i$, where m_i is the message of P_i. Also, each player P_i computes $\sigma_i = \{\sigma_i(1), \sigma_i(2), \ldots, \sigma_i(\mathsf{N})\}$, where σ_i includes the identity of player P_i and a proof of valid formatting of V_i. Then P_i publishes both the vectors V_i and σ_i. Our model assumes the pair $(V_i(w), \sigma_i(w))$ for each position w as a single packet, where $V_i(w)$ is the packet content and $\sigma_i(w)$ becomes a part of *meta* field — and the tag field

[1]↑Since we are mainly interested in the anonymity property, we don't need to model the part of the protocol where the protocol parties reconstructs the keys in case of a failure. But it is easy to extend M_{DC} to include that step by adding one more round to the current model.

[2]↑The anonymity game does not include multiple sessions. Also, in our model all the N players participate in a protocol run.

contains the value w. For each position w player P_i generates one such packet, and publishes the packets to all other players.

- *verify* and *extract* are local computations after a party P_i receives packets from all other parties.

Although the protocol model assumes that the adversary can not read the contents of any packet, here we model Π_{DC} along with its cryptographic primitives to demonstrate the expressiveness of our model. Alternatively, to get rid of all the cryptographic primitives, the parties can send a dummy message ($= 0$) whenever $V_i(w) = W_i(w)$, and the actual message m_i whenever $V_i(w) \neq W_i(w)$.

As per our anonymity definition in Section 2.1, we assume that up to $(\mathsf{N} - 2)$ users can be compromised, which necessarily makes up to $(\mathsf{N} - 2)$ protocol parties compromised. The adversary chooses two challenge users, and one of them sends the challenge message depending on the challenge bit b. All other $(N - 1)$ users also participate in the user coordination process and send shares for the challenge message.

In M_{DC} we model Π_{DC} as a two round protocol. The challenger sets the initial configuration of the Petri net with the messages to be sent by each party. In the first round, each party P_i sends two kinds of packets: (1) publishes the public key y_i and (2) sends share of the public-private keypair $(x_{i,j}, y_{i,j})$ to P_j for all $j \neq i$. In the second round, each party P_i publishes N packets: one packet for each position, only one of them contains his own message. After second round, every party receives packets from every other party, and then does local computations to verify and extract the original messages.

For Π_{DC}, we do not actually need a separate recipient R in Π_{DC}, if we make $\mathcal{R} = \mathsf{P}$. But, to be consistent with M, in M_{DC} we keep a separate recipient. In the second round whenever a party P_i publishes a packet, P_i also sends a copy to R. This easily models the fact that the adversary knows whenever a message is published, but avoids the complication of modeling a subset of compromised recipients.

The *meta* fields of the tokens contains the following subfields: (1) stage, (2) sigma. The subfield *stage* can have three possible values identifying three possible cases: (1) public key distribution, (2) share of the public-private key pair, (3) message. Using the *stage* subfield, any party in the protocol recognizes if the message is part of *keydist* messages, or part of

post messages. When the value of *stage* is *message*, the user posts $V_i(w)$ for the msg field, and w for the tag field in the petri net token. The *sigma* subfield in the *meta* field includes the identity of the sender and a proof of computation whenever necessary; *sigma* subfield helps in the *verify* stage, we avoid the details here.

If we want to analyze the user distribution for Π_{DC}, we do not count the first round since it is used only for the key exchange and no user message is sent. Note that, if we get rid of the cryptographic primitives, we do not require the first round. If we assume that all the users are ready with their messages at the beginning, the latency overhead of Π_{DC} is 1, and the bandwidth overhead is $\geq (N-1)$ per message.

A.2.2 Modeling Tor

This section demonstrates that onion routing protocols like Tor can be easily modeled using our Petri net model M. We want to stress that we focus on sender anonymity game against a global passive adversary, and hence, we do shall not model any sophisticated features like hidden services, congestion control etc.

Since Tor does not operate in rounds, embedding it into our model is not straight forward. Suppose, a Tor node takes at least x milliseconds to process a message when it receives a message, and it takes at least y milliseconds for a message to travel from one node to the next node over a network link. Then we define *one round* as $x + y$ milliseconds. We assume a perfect condition where each node takes exactly equal computation time for one message, and each link has exactly same delay. In the real world, delays and computation times are less stable, but can be estimated by an adversary. Instead of analyzing this, we instead allow the messages to remain within the node for the respective time. Tor nodes and recipients are separate entities and hence, $\mathcal{S}$, P and R are mutually exclusive.

Whenever a Tor node receives a message, the node immediately processes and forwards that message to the next node or recipient. We can either model the latency overhead ℓ of Tor by estimating the time messages spend within the network that exceeds the (minimal) round length $x + y$ from above, or we set it to the number of hops, i.e., $\ell = 3$. In either case, we assume that ℓ does not increase with η and thus get a latency overhead $\ell \in O(1)$. For

analyzing Tor with a variable number h of hops, we can instead set $\ell = h$. When a party P_i receives a message, T_{P_i} can retrieve the next hop from the meta field of the message. Since Tor does not add any noise messages, the bandwidth overhead is $\beta = 0$.

A.3 Visual 3D representations of the results

In the work, we focus on lower-bound results for strong anonymity (or negligible δ values). However, our key Theorems 3.2.1, 3.3.1, 3.4.1 and 3.5.1 also offer lower bounds for non-negligilable δ values, which can be of interest to several AC protocols. On our project webpage [52], we visualize these lower bounds using interactive 3D surface plots. In particular, we plot the adversarial advantage $\delta \in [0, 1]$ as a function of β and ℓ. We encourage the readers to interact with these plots for an intuitive understanding of our results for non-negligilbe δ values.

Here, in Figures A.1 to A.4, we present and analyze four snapshots of those lower bound plots for the number of users $N = 10000$. The x-axis represents latency ℓ (ranging from 0 to 200), and the y-axis bandwidth overhead β (ranging from 0.0 to 0.04). But in Figure A.3 and Figure A.4, the y-axis actually represents total bandwidth $p = p + \beta$ as in Theorem 3.4.2. For curves in Figure A.2 and Figure A.4 we chose $\mathsf{K} = 100$ as the number of total parties and $\mathsf{c} = 20$ compromised parties. In each plot, the dark blue region indicates the possibility of obtaining strong anonymity. For any point (x, y) outside those regions, strong anonymity is not possible. For example, as shown in Figure A.1, for $\ell = 100$ the bandwidth overhead β has to be at least 0.01 to expect strong anonymity.

A derived δ lower bound for the non-compromising adversary is also a valid lower bound for a (partially) compromising adversary. For some edge cases (e.g., when ℓ is close to N and β is close to 0), due to some approximations employed in the compromising adversaries scenario, the non-compromising adversary lower bound is actually tighter than the compromising adversaries lower bound. Therefore, in Figure A.4, while plotting the 3D graph for a partially compromising adversary scenario, we have used the maximum of the lower bounds on δ for compromising adversary and non-compromising adversary.

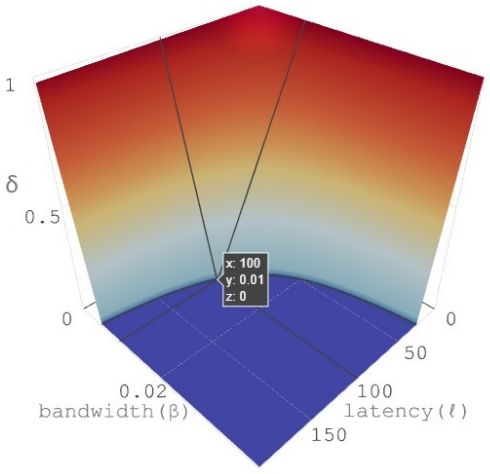


Figure A.1. Synchronized User Distribution with Non-compromising Adversaries. $z = 1 - f_\beta(\ell)$, where $f_\beta(x) = \min(1, ((x + \beta \mathsf{N} x)/(\mathsf{N} - 1)))$.

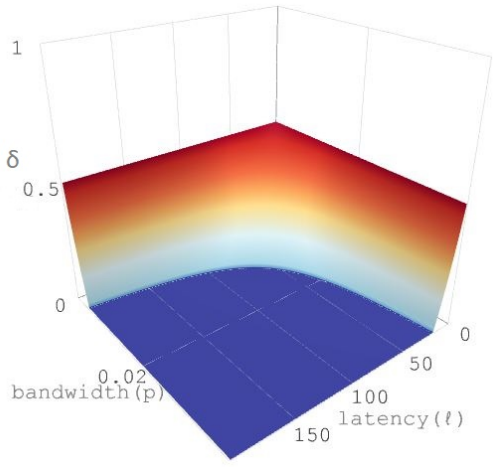


Figure A.3. Unsynchronized User Distribution with Non-compromising Adversaries. $z = 1 - \left(\frac{1}{2} + f_p(\ell)\right)$, where $f_p(x) = \min(1/2,\ 1 - (1-p)^x)$.

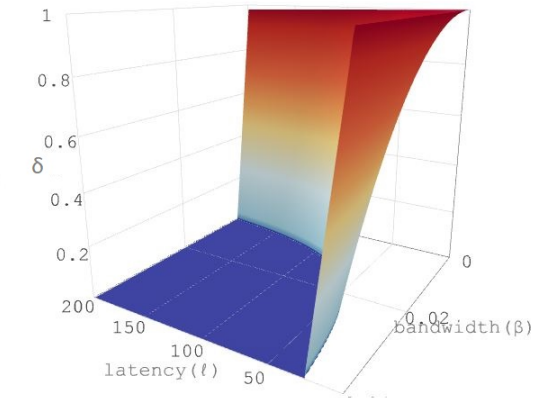


Figure A.2. Synchronized User Distribution with Partially compromising Adversaries. Total protocol parties $\mathsf{K} = 100$, number of compromised parties $\mathsf{c} = 20$. $z = 1 - [1 - \binom{\mathsf{c}}{\ell}/\binom{\mathsf{K}}{\ell}] f_\beta(\ell)$ for $\ell \leq \mathsf{c}$, $z = 1 - [1 - 1/\binom{\mathsf{K}}{\mathsf{c}}] f_\beta(\mathsf{c}) - f_\beta(\ell - \mathsf{c})$ otherwise.

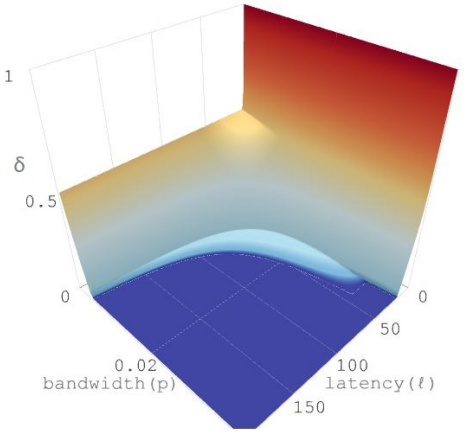


Figure A.4. Unsynchronized User Distribution with Partially compromising Adversaries. Total number of protocol parties $\mathsf{K} = 100$, number of compromised parties $\mathsf{c} = 20$. $z = 1 - [1 - \binom{\mathsf{c}}{\ell}/\binom{\mathsf{K}}{\ell}][\frac{1}{2} + f_p(\ell)]$ for $\ell \leq \mathsf{c}$, $z = (1 - [1 - 1/\binom{\mathsf{K}}{\mathsf{c}}][1/2 + f_p(\mathsf{c})]) \cdot (1 - [1/2 + f_p(\ell - \mathsf{c})])$ otherwise. We set $z = \max(z, 1 - (1/2 + f_p(\ell)))$

For the chosen c and K, the plots in Figures A.1 and A.2 are almost identical as the ℓ and β factors contribute more to anonymity than the compromised parties can affect it. If we instead compare Figure A.3 with Figure A.4, the effect of compromisation is noticeable: the dark blue region in Figure A.4 is much smaller than that in Figure A.3. Also, we can see a steep wall in Figure A.4 for $\ell \leq \mathsf{c} = 20$, demonstrating that providing anonymity becomes difficult when $\ell < \mathsf{c}$; however, for $\ell > \mathsf{c}$, the effect of compromisation is less noticeable.

VITA

Debajyoti Das (he/him/his) is a Ph.D. candidate in Computer Science department at Purdue University. His research interests lie at the intersection of applied cryptography and privacy – He designs, implements and analyzes privacy-preserving systems.

Debajyoti received his Bachelor of Technology from Indian Institute of Technology Hyderabad in 2013. Then he worked as a software engineer at Microsoft India Development Center in Hyderabad for 2 years before joining Purdue in 2015. He is fluent in three languages (Bengali, Hindi, and English), and can speak basic Spanish.

PUBLICATION(S)

1. Debajyoti Das[3], Easwar V. Mangipudi[3], Aniket Kate:
   OrgAn: Organizational Anonymity with Low Latency.
   (Under Submission.)

2. Debajyoti Das, Sebastian Meiser, Esfandiar Mohammadi, Aniket Kate:
   Streams:Provably Secure Network Anonymity at Scale.
   (Under Submission.)

3. Mic Bowman, Debajyoti Das, Avradip Mandal, Hart Montgomery:
   On Elapsed Time Consensus Protocols.
   Available at: https://eprint.iacr.org/2021/086.pdf

4. Debajyoti Das, Sebastian Meiser, Esfandiar Mohammadi, Aniket Kate:
   Comprehensive Anonymity Trilemma: User Coordination is not enough.
   20th Privacy Enhancing Technologies Symposium, 2020.

5. Debajyoti Das, Sebastian Meiser, Esfandiar Mohammadi, Aniket Kate:
   Anonymity Trilemma: Strong Anonymity, Low Bandwidth Overhead, Low Latency – Choose Two.
   39th IEEE Symposium on Security and Privacy, 2018.

6. David Chaum, Debajyoti Das, Farid Javani, Aniket Kate, Anna Krasnova, Joeri de Ruiter, and Alan T. Sherman:
   cMix: Mixing with Minimal Real-Time Asymmetric Cryptographic Operations.
   15th International Conference on Applied Cryptography and Network Security, 2017.

[3]↑Both authors contributed equally to that paper.