Anonymous Router Network

by

Anthony S. Flath

A thesis submitted in partial fulfillment
of the requirements for the degree of
Bachelor of Arts
(Honors Computer Science)
in The University of Michigan
2026

Reader Committee:
Professor Ke Wu, Advisor
Professor Paul Grubbs

Anthony S. Flath
aflath@umich.edu

DEDICATION

To my partner, Sarah.

ACKNOWLEDGEMENTS

This work would not be possible without the advice and guidance of Ke Wu, for which I am eternally grateful. Thank you for helping me begin my academic journey.

I would also like to extend my appreciation to Paul Grubbs for being a secondary reader for this thesis; this work would not have been able to begin without him.

Thank you to Brent Sauve. The encouragement and help in furthering my interest in the field led me to this work.

Finally, I am extremely thankful for Sarah Stager for listening to me talk about routers for six months. Additionally, thank you for suggesting improvements to my writing. Your support throughout this entire process has been invaluable.

TABLE OF CONTENTS

List of Tables

ABSTRACT

Anonymous networking is an essential tool for internet privacy. Most constructions consist of trusting some threshold of honest actors to ensure anonymity (e.g. Tor, mixnets, among others). Recent works have proposed anonymous routers in which a single untrusted router receieves and sends messages without knowing any message paths. In particular, the work of Das and Park on sPAR (somewhat Practical Anonymous Router) explored how to implement a router using multi-party fully homomorphic encryption schemes that are practical to implement in real use cases. However, their design had performance limitations when routing messages.

To offload the computation costs, we propose a network composed of similarly constructed routers in which messages can be transmitted at a faster rate. While we do not necessarily achieve faster benchmarks than previous works, we lay important groundwork for future work. Specifically, we show that offloading computation costs without sacrificing security is possible and easily implementable.

1 Introduction

Anonymous communication is an essential tool of the modern age of the internet, in which users can communicate without revealing their identity. The field includes many different protocols including onion routing [9], Mixnets [20], and DC-nets [16]. Many of these systems depend on a certain portion of honest actors for security and/or direct interactions between communicating clients. An implication of this is that security is based on a probabilistic model of how many malicious actors are within the system.

Thus, trust is put into the system based on the assumption that there are *enough* honest actors. This leads to a fundamental flaw: given enough resources, an adversary can corrupt as many actors as needed for their purposes. Though there are attempts to prevent this [10], it is an ongoing issue given the nature of the security.

In contrast to the above security notions, a new form of communication was proposed by Shi and Wu [21] called Non-Interactive Anonymous Router (NIAR), in which anonymity is guaranteed through one central router such that clients do not interact. Unlike previous models, this does not require a threshold of honest parties to ensure anonymity. Instead, it provides anonymity even when the router is malicious.

Given that this is a recent novel idea, there has been little research into implementation of this NIAR-form of routing, but published works that do realize the idea suffer from severe limitations. Performance has been slow on instantiations due to the expensive cryptographic building blocks and a bottleneck of one machine. Additionally, reliability is only guaranteed based on the work of a single router. This leaves some open questions:

1. How can efficiency be improved without sacrificing security?
2. Is it possible to eliminate the reliance on the participation of all parties?

1.1 Existing Works

NIAR shows that this form of anonymous communication is theoretically possible, but Shi and Wu's work focuses on concepts rather than implementation. Thus, the real-life implications of deployment are not entirely clear. It is clear, however, that protocols aiming to work on an untrusted server must realize oblivious permutation.

We closely follow the work of Das and Park on somewhat Practical Anonymous Router (sPAR) [8], in which they implement a protocol that realizes a central anonymous routing scheme with a relaxed notion of the non-interactive property as proposed in NIAR. They instantiate the concept utilizing fully homomorphic encryption (FHE) for multiple parties to perform operations on ciphertexts [17] [13]. As a result, the router can obliviously map incoming messages to their destination without leaking routing information. Their work is practical to implement due to the wide breadth of research and open source libraries on FHE.

At a high level, their oblivious mapping mechanism relies on the router obliviously placing ciphertexts into a matrix held by the router. However, their construction lacks clarity with respect to the operations done on ciphertexts. More specifically, it does not

state the types of ciphertexts being operated on when mapping. In addition, the size of said matrix scales linearly with the number of users within the system. Thus, the number of operations needed to place messages also scales with the number of users. This means that as the number of users increases, the speed of the protocol is drastically increased.

1.2 Our Results

While the work of sPAR is functional, we aim to speed up the protocol as the number of users grows while keeping the same security guarantees. It intuitively makes sense to attempt to decrease the amount of work that routers do. The reason this was not done before was to ensure that all messages could fit into a router's matrix. In addition, we assume that every router is corrupted.

Our results show that it is possible to distribute the work done to multiple routers without sacrificing security. More specifically, our implementation achieves near-linear scaling of the amount of work done with respect to the number of routers. Though we only give benchmark for a small number of users (128 and less), it is possible to increase the number of users as we discuss in Section 6. Outside of performance improvements, in a deployment scenario, adding more routers allows some messages to be transmitted even if some routers fail to continue operating.

To situate our construction in context, it is helpful to first consider a trivial implementation in which each user sends its message to a router of the user's choice. However, that router knows that one of the messages it sends originates from said user. Thus, the router knows that the transmitted message comes from a subset of users, i.e. the users that picked the router to transmit their message.

Given this, we propose and implement a system in which users send encrypted messages to every router, and in which only one receives the true encrypted message. The other messages do not contain meaningful information and do not meaningfully affect the oblivious mapping mechanism. Thus, given a secure encryption scheme, this results in routers not being able to distinguish between meaningful and meaningless ciphertexts, meaning they cannot correlate a transmitted message with a subset of senders.

Though we closely follow sPAR's placing algorithm, we depart from Das and Park's construction due to the lack of clarity of the operations performed. However, we show that a trivial implementation is sufficient to show the stated goals of distributing work to routers. We also discuss how including their construction would affect computation time and how it can be further improved.

2 Preliminaries

2.1 Communication Model

We consider a communication system composed of a finite set of senders, routers, and receivers for transmitting n messages from n senders to n' receivers through k routers. We only consider the scenario in which $n' = n$, though the process could be generalized to

$n' \neq n$. As detailed in Table 1, we denote a sender as $i \in [n]$ for some $i \in \{1, \ldots, n\}$. Similarly, we denote a receiver as $\ell \in [n]$ and a router as $j \in [k]$. Although a sender can act as a receiver and vice versa in implementation, we will consider them as distinct identities. We refer to any singular sender or receiver as a user.

In addition, each router holds an array of $p \cdot n$ values to aid in our oblivious placing mechanism such that $n \cdot p$ is a multiple of 2. We call p the bin multiplier, i.e. the value which we multiply by n to calculate the size of a router's placing array.

We denote the multiset of messages as $\mathcal{M}$ such that for all messages sent, $\{m_i\}_{i \in [n]}$, $m_i \in \mathcal{M}$. Following this, π is denoted as the mapping of messages sent by senders to receivers such that for the ith sender, the message mapping is (i, π_i, m_i). In other words, π is the set of all pathways that messages travel during a round of communication. For example, $\pi_1 = 4$ represents sender 1 sends a message to receiver 4. We note that a receiver can be sent more than one message, so we denote π_ℓ^{-1} as the set of all senders that send to the receiver ℓ.

n	total number of senders/receivers
k	total number of routers
p	bin multiplier
i	some sender s.t. $i \in [n]$
ℓ	some receiver s.t. $\ell \in [n]$
j	some router s.t. $j \in [k]$
$\mathcal{M}$	multiset of sent messages
π	set of message mappings
$[c]$	set of values $\{1, \ldots, c\}$ for integer c

Table 1: Notation used in defining the system.

There exists a pairwise channel between each sender and router, and a pairwise channel between each router and receiver. There is no communication channel available between any pair of senders and receivers or any pair of routers.

The system consists of a one-time setup protocol followed by a number of rounds of communication. Each round consists of senders transmitting encrypted messages through the previously mentioned channel to each router. Following this, the router performs computations on the encrypted messages according to the mapping to create some transformed ciphertext. After decryption, the routers will then forward the decrypted messages to all receivers. Senders and receivers do not directly interact with one another during any round, but rather through the intermediate routers.

We assume that an arbitrary number of routers are colluding, and there are at least two senders and two receivers that are acting independently and honestly. All information shared with colluding parties is done outside of the communication system defined here. We first propose a construction that assumes that all parties are honest-but-curious; however, we discuss adjustments that consider active attacks.

We first propose a version that provides sender anonymity, meaning that messages sent

through the network cannot be traced to their senders and all decrypted messages are known to an adversary. Following this, adaptations to the protocol will be given in order to provide receiver anonymity. This means, in addition to sender anonymity restrictions, that an adversary cannot determine intended recipients of messages sent through the network and only knows the decrypted messages it receives.

2.2 Syntax

An Anonymous Router Network (ARN) is cryptographic scheme consisting of the following algorithms:

- $(\mathsf{pk}, \{\mathsf{ek}_i\}_{i\in[n]}, \{\mathsf{rk}_\ell\}_{\ell\in[n]}) \leftarrow \mathbf{Setup}(1^\lambda, n, k, p)$: For the security parameter 1^λ, the trusted **Setup** algorithm takes the number of senders/receivers n and the number of routers k. It outputs a public key pk, a sender key for each sender denoted $\{\mathsf{ek}_i\}_{i\in[n]}$, and a receiver key for each receiver denoted $\{\mathsf{rk}_\ell\}_{\ell\in[n]}$.
- $(\{c_{i,j}\}_{j\in[k]}) \leftarrow \mathbf{Enc}(\mathsf{pk}, m_i, \pi_i)$: The sender i uses **Enc** in which its message m_i is encrypted according to the mapping π_i with the public key pk and outputs ciphertexts $\{c_{i,j}\}_{j\in[k]}$.
- $(C_j) \leftarrow \mathbf{Rte}(\widehat{\mathsf{pk}}, \{c_{i,j}\}_{i\in[n]})$: The router j uses the **Rte** algorithm which takes all ciphertexts sent to it by n senders denoted as $\{c_{i,j}\}_{i\in[n]}$. It outputs a $p \cdot n$ array C_j such that it contains the transformed input ciphertexts.
- $(\mathcal{M}_j) \leftarrow \mathbf{Dec}(\{\mathsf{ek}_i\}_{i\in[n]}, \{\mathsf{rk}_\ell\}_{\ell\in[n]}, C_j)$: The **Dec** algorithm takes all sender keys denoted as $\{\mathsf{ek}_i\}_{i\in[n]}$, all receiver keys denoted as $\{\mathsf{rk}_\ell\}_{\ell\in[n]}$, and an array holding all transformed ciphertexts C_j for the jth router. The algorithm outputs the multiset of non-zero decrypted messages $\mathcal{M}_j$ from C_j.

Note that the trusted setup must be done collectively, i.e. all participants within the system must agree on some pk. In addition, n, k, and p are known a priori to compute **Setup**. Following this, senders can communicate with receivers $\mathsf{poly}(1^\lambda)$ timesteps.

2.3 Security Requirements

Following the framework from NIAR [21] and similar notions of security [3], we define our security requirements and definitions. We emphasize that the corrupted party is honest but curious. In defining our security, we denote the set of honest users as $\mathcal{H}$ and the set of corrupt users as $\mathcal{K}$.

Defining leakage known to corrupt parties. When some senders and receivers are corrupt, the adversary may learn some information about the message mapping π and $\mathcal{M} \coloneqq \{\forall i \in \mathcal{H}_S : m_i\}$, the multiset of all honest senders' messages. The function Leak is defined in the following sections and is used to describe the leaked information when considering the following security notions.

Sender Anonymity We define sender anonymity as the adversary knowing the content of every message and the only knowledge leaked about π being from corrupt senders. In other words, the adversary only knows the origin of messages when sent from corrupted senders. More formally, we define the leakage of the adversary for sender anonymity as the following:

$$\mathsf{Leak}_{\mathsf{SA}}(\pi, \mathcal{M}, \mathcal{K}_S, \mathcal{K}_R) \coloneqq \{\forall i \in \mathcal{K}_S : (i, \pi_i, m_i)\} \cup \mathcal{M},$$

Recipient Anonymity Recipient anonymity, in addition to the restrictions in sender anonymity, is defined as the adversary being unable to determine the destination of any message sent by honest senders, and the content of any message it does not receive. More formally, given $\pi^{-1}_{\mathcal{K}_R}$ is the set of senders that send to a corrupted receiver, we define the leakage of the adversary for recipient anonymity as the following:

$$\mathsf{Leak}_{\mathsf{RA}}(\pi, \mathcal{M}, \mathcal{K}_S, \mathcal{K}_R) \coloneqq \{\forall i \in \mathcal{K}_S \cap \pi^{-1}_{\mathcal{K}_R} : (i, \pi_i, m_i)\} \cup \mathcal{M}_{\mathsf{HK}}.$$

ARN Experiment. The ARN experiment is parametrized by bit $b \in \{0, 1\}$, an adversary $\mathcal{A}$, and a security parameter λ. We denote $\mathcal{T}_t$ as the set of information known by the adversary in the timesteps $\{1, \ldots, t-1\}$. Note that the adversary $\mathcal{A}$ corrupts all routers, a set of senders $\mathcal{K}_S \subseteq [n]$, and a set of recipients $\mathcal{K}_R \subseteq [n]$. Let $\mathcal{H}_S = [n] \setminus \mathcal{K}_S$ be the set of honest senders and $\mathcal{H}_R = [n] \setminus \mathcal{K}_R$ be the set of honest receivers. We require there to be at least two honest senders and two honest recipients. Consider the following experiment:

$\mathsf{ARN\text{-}Expt}^{b,\mathcal{A}}(1^\lambda)$

- $n, k, p, \mathcal{K}_S, \mathcal{K}_R \leftarrow \mathcal{A}(1^\lambda)$
- $\widehat{\mathsf{pk}}, \{\mathsf{ek}_i\}_{i\in[n]}, \{\mathsf{rk}_\ell\}_{\ell\in[n]} \leftarrow \mathbf{Setup}(1^\lambda, n, k)$
- For $t = 1, 2, \ldots, \mathsf{poly}(1^\lambda)$
  - $\{\{(i, \pi_i^b, m_i^b)\}_{i\in\mathcal{H}_S}\}_{b\in\{0,1\}}, \{\{c_{i,j}\}_{j\in[k]}\}_{i\in\mathcal{K}_S} \leftarrow \mathcal{A}(\widehat{\mathsf{pk}}, \{\mathsf{ek}_i\}_{i\in\mathcal{K}_S}, \{\mathsf{rk}_\ell\}_{\ell\in\mathcal{K}_R}, \mathcal{T}_t)$
  - for $i \in \mathcal{H}_S$, $\{c_{i,j}\}_{j\in[k]} \leftarrow \mathbf{Enc}(\widehat{\mathsf{pk}}, m_i^b, \pi_i^b)$
  - $\forall j \in [k] : C_j \leftarrow \mathbf{Rte}(\widehat{\mathsf{pk}}, \{c_{i,j}\}_{i\in[n]})$
  - $\forall j \in [k] : \mathcal{M}_j \leftarrow \mathbf{Dec}(\{\mathsf{ek}_i\}_{i\in[n]}, \{\mathsf{rk}_\ell\}_{\ell\in[n]}, C_j)$

We say that $\mathcal{A}$ is admissible if and only if in both worlds, i.e. $b = 0$ or $b = 1$, it guarantees with probability 1 that for each round

1. $\mathsf{Leak}(\pi^0, \mathcal{M}, \mathcal{K}_S, \mathcal{K}_R) = \mathsf{Leak}(\pi^1, \mathcal{M}, \mathcal{K}_S, \mathcal{K}_R)$,
2. $\mathcal{M}^0 = \mathcal{M}^1$, i.e. the content of every message sent in both worlds by honest senders is the same, and

3. for any $\ell \in \mathcal{K}_R \cap \{\pi_i\}_{i\in\mathcal{H}_S}$, $m^0_{i,\ell} = m^1_{i,\ell}$, i.e. every corrupted receiver must receive the same message in both worlds when receiving from honest senders.

Definition 1 (Sender Anonymity)**.** *The protocol provides sender anonymity iff when* $\mathsf{Leak} := \mathsf{Leak}_{\mathsf{SA}}$*, for any probabilistic polynomial time (p.p.t.) adversary* $\mathcal{A}$ *and some negligible* ϵ*,*

$$|\Pr[\mathit{ARN\text{-}Expt}^{0,\mathcal{A}}_{\mathsf{SA}}(1^\lambda) = 1] - \Pr[\mathit{ARN\text{-}Expt}^{1,\mathcal{A}}_{\mathsf{SA}}(1^\lambda) = 1]| \leq \epsilon(\lambda).$$

Definition 2 (Recipient Anonymity)**.** *The protocol provides receiver anonymity iff when* $\mathsf{Leak} := \mathsf{Leak}_{\mathsf{RA}}$*, for any p.p.t. adversary* $\mathcal{A}$ *and some negligible* ϵ*,*

$$|\Pr[\mathit{ARN\text{-}Expt}^{0,\mathcal{A}}_{\mathsf{RA}}(1^\lambda) = 1] - \Pr[\mathit{ARN\text{-}Expt}^{1,\mathcal{A}}_{\mathsf{RA}}(1^\lambda) = 1]| \leq \epsilon(\lambda).$$

3 Building Blocks

To implement the anonymous router network, we introduce a multi-party fully homomorphic encryption scheme based on the ring learning with errors problem (RLWE). We also discuss a placing mechanism in which the router obliviously places incoming ciphertexts.

3.1 Multi-Party FHE

To instantiate our protocol, we will use the multi-party FHE technique from [18] to perform calculations on ciphertexts encrypted by multiple parties' keys [14][2]. The goal is to be able to encrypt a message that can only be decrypted with all users' secret key. The public key must not reveal any information about any single secret key and a ciphertext encrypted under the public key must not reveal any information about the underlying message.

This requires a one-time setup creating a shared public key followed by computations among a fixed number of parties. We define operations used in multi-party FHE schemes:

- $(\widehat{\mathsf{pk}}, \{\mathsf{ek}_i\}_{i\in[n]}, \{\mathsf{rk}_\ell\}_{\ell\in[n]}) \leftarrow \mathbf{Setup}(1^\lambda, n, k)$: For the security parameter 1^λ, the **Setup** algorithm generates secret keys $\{\mathsf{ek}_i\}_{i\in[n]}$ for senders and secret keys $\{\mathsf{rk}_\ell\}_{\ell\in[n]}$ for receivers. The public key $\widehat{\mathsf{pk}}$ is generated with its corresponding secret key $\widehat{\mathsf{sk}} = \sum_{i=0}^{n} \mathsf{ek}_i + \sum_{\ell=0}^{n} \mathsf{rk}_\ell$ not known to any single participant within the scheme.
- $(c) \leftarrow \mathbf{Enc}(\widehat{\mathsf{pk}}, m)$: Message m gets encrypted with the public key $\widehat{\mathsf{pk}}$ to receive the encrypted ciphertext c.
- $(c^*) \leftarrow \mathbf{Eval}(F, c_1, c_2)$: Computation is performed on two ciphertexts c_1, c_2 according to the function F.
- $(m^*) \leftarrow \mathbf{Dec}(\{\mathsf{ek}_i\}_{i\in[n]}, \{\mathsf{rk}_\ell\}_{\ell\in[n]}, c^*)$: The **Dec** algorithm takes all secret keys to decrypt a ciphertext to output the message m^*.

An FHE scheme allows computation on ciphertexts [11]. More specifically, we will be using homomorphic addition and homomorphic multiplication.

Homomorphic addition Let $c_1 = \mathbf{Enc}(\widehat{\mathsf{pk}}, m_1)$ and $c_2 = \mathbf{Enc}(\widehat{\mathsf{pk}}, m_2)$ for key k. We define homomorphic addition as F^+ such that

$$m_1 + m_2 = \mathbf{Dec}(\{\mathsf{ek}_i\}_{i\in[n]}, \{\mathsf{rk}_\ell\}_{\ell\in[n]}, \mathbf{Eval}(F^+, c_1, c_2)).$$

Homomorphic multiplication Let $c_1 = \mathbf{Enc}(m_1)$ and $c_2 = \mathbf{Enc}(m_2)$ for key k. We define homomorphic multiplication as $F^\times$ such that

$$m_1 \cdot m_2 = \mathbf{Dec}(\{\mathsf{ek}_i\}_{i\in[n]}, \{\mathsf{rk}_\ell\}_{\ell\in[n]}, \mathbf{Eval}(F^\times, c_1, c_2)).$$

Correctness We require with probability 1 that

1. the above operations are supported, and
2. given m, $\widehat{\mathsf{pk}}$, $\{d_i\}_{i\in[n]}$, $\{d_\ell\}_{\ell\in[n]}$, and $c \coloneqq \mathbf{Enc}(\widehat{\mathsf{pk}}, m)$, then $m = \mathbf{Dec}(\{d_i\}_{i\in[n]}, \{d_\ell\}_{\ell\in[n]}, c)$.

Definition 3 (Multi-Party Semantic Security)**.** *Given a security parameter 1^λ and a valid FHE scheme, a multi-party FHE protocol provides semantic security iff for $m_1, m_2 \in \mathbb{R}_t$ and any p.p.t. adversary $\mathcal{A}$ with knowledge of $2n-1$ secret keys, its views of $\boldsymbol{Enc}(\widehat{\mathsf{pk}}, m_1)$ and $\boldsymbol{Enc}(\widehat{\mathsf{pk}}, m_2)$ are computationally indistinguishable.*

3.2 FHE from Ring Learning With Errors

To achieve the operations listed above, we adopt a scheme [5][4][12] whose security is based on the RLWE problem [15].

Notation We denote λ as the security parameter. Let $\mathcal{R} = \mathbb{Z}[x]/(x^N + 1)$ and $\mathcal{R}_q = (\mathcal{R} \bmod q)$ for positive integers q and N. Table 2 summarizes our notation.

λ	security parameter
N	polynomial degree
q	ciphertext modulus
t	plaintext modulus
$\mathcal{R}$	$\mathbb{Z}[x]/(x^N + 1)$, $N \in \mathbb{N}$
$\mathcal{R}_q$	$\mathbb{Z}[x]/(x^N + 1) \mod q$, $q, N \in \mathbb{N}$
$\xleftarrow{\$}$	sample from distribution

Table 2: Notation used in encryption schemes.

Definition 4 (Ring Learning With Errors)**.** *Given the security parameter λ is a power of 2, for a fixed secret element $s \in R_q$ and a distribution $\chi = \chi(\lambda)$ over $\mathcal{R}$, the decisional RLWE problems states the distribution outputting $(a, a \cdot s + e)$, where a and e are respectively chosen uniformly at random from R_q and χ and the distribution outputting (a, u) chosen uniformly at random from $\mathcal{R}_q^2$ are computationally indistinguishable.*

Ciphertexts For plaintext modulus t and ciphertext modulus q where $t \ll q$, let $\Delta = \lfloor \frac{q}{t} \rceil$, $m \in \mathcal{R}_t$ be a polynomial message, and $s \in \mathcal{R}_q$ be a secret key.

- An RLWE ciphertext, denoted by $\mathsf{RLWE}_s(m)$, is defined as $c = (a, b) \in \mathcal{R}_q^2$ such that $b = a \cdot s + \Delta \cdot m + e$ for some small error e.
- Let β be a fixed base and $\alpha = \log_\beta(q)$. We define the gadget vector as $g = (1, \beta, \ldots, \beta^{\alpha-1})$ and the gadget matrix as $G = I_2 \otimes g$ where I_2 is an identity matrix. An RGSW ciphertext [6] is defined as a matrix $C = H + \Delta \cdot m \cdot G = \mathsf{RGSW}_s(m) \in \mathcal{R}_q^{2 \times 2\alpha}$ where H is a matrix such that each column is $\mathsf{RLWE}_s(0)$.

Homomorphic addition. Let $c_1 = (a_1, b_1)$ and $c_2 = (a_2, b_2)$ be two RLWE ciphertexts. We define the addition of the two ciphertexts as $c_1 + c_2 = (a_1 + a_2, b_1 + b_2)$.

External product. Homomorphic multiplication between RGSW and RLWE ciphertexts is named external product in the literature. More specifically, for messages $m_0, m_1 \in \mathcal{R}_t$ and secret key $s \in \mathcal{R}_q$, external product results in $\mathsf{RLWE}_s(m_0) \cdot \mathsf{RGSW}_s(m_1) = \mathsf{RLWE}_s(m_0 \cdot m_1)$. We point to [6] for more detail.

3.3 RLWE Multi-Party FHE

We use multi-party FHE based on RLWE in order to build our protocol [18]. To achieve the goals stated above, we rely on [19] for IND-CPA security of the instantiation of the scheme. Specifically, the RLWE-based scheme achieves these goals assuming the RLWE assumption defined in Definition 4.

The following algorithms are parameterized by $\mathsf{pp} \coloneqq \{N, q, t, \chi, \chi'\}$, which is known by all parties, where N is the polynomial degree, q is the ciphertext modulus, t is the plaintext modulus, χ is the secret distribution such that $\forall x \leftarrow \chi$, $x \in \mathcal{R}_q$, and χ' is the error distribution such that $\forall e \leftarrow \chi'$, $e < \frac{\Delta}{2}$. Similar to using standard multi-party computation techniques to compute the setup, we can do the same for pp.

- $(\widehat{\mathsf{pk}}, \{\mathsf{ek}_i\}_{i \in [n]}, \{\mathsf{rk}_\ell\}_{\ell \in [n]}) \leftarrow \mathbf{Setup}_{\mathsf{pp}}(1^\lambda, n, k)$: All players agree on some $a \xleftarrow{\$} \mathcal{R}_q$ as some random parameter.
  - All senders and receivers generate their secret key $\mathsf{uk} \xleftarrow{\$} \chi$ and calculate their local public key $\mathsf{pk} = (a \cdot \mathsf{uk} + e) \in \mathcal{R}_q$ for the error sample $e \xleftarrow{\$} \chi'$.

– All parties calculate $\widehat{\mathsf{pk}} = (a, \sum_{i=1}^{n} \mathsf{pk}_i + \sum_{\ell=1}^{n} \mathsf{pk}_\ell) \in \mathcal{R}_q$. We denote the public key's corresponding secret key as $\widehat{\mathsf{sk}} = \sum_{i=1}^{n} \mathsf{ek}_i + \sum_{\ell=1}^{n} \mathsf{rk}_\ell$ such that $\widehat{\mathsf{sk}} \in \mathcal{R}_q$ and $a \cdot \widehat{\mathsf{sk}} + \widehat{\mathsf{pk}} \approx 0 \mod q$. No players know the corresponding secret key.
– Output $\widehat{\mathsf{pk}}, \{\mathsf{ek}_i\}_{i\in[n]}, \{\mathsf{rk}_\ell\}_{\ell\in[n]}$

- $(c_i) \leftarrow \mathbf{Enc}_{\mathsf{pp}}(\widehat{\mathsf{pk}}, m_i)$: The ith sender encrypts message m_i with $\widehat{\mathsf{pk}} = (a, b)$ to receive the ciphertexts $\{c_{i,j}\}_{j\in[k]}$. For samples $u \xleftarrow{\$} \chi$ and $e_1, e_2 \xleftarrow{\$} \chi'$, $c_i = (u \cdot a + e_1, u \cdot b + \Delta m + e_2)$. We note that this returns $\mathsf{RLWE}_{\widehat{\mathsf{pk}}}(m_i)$; however, it is possible to return $\mathsf{RGSW}_{\widehat{\mathsf{pk}}}(m_i)$ as described in Section 3.2.

- $(c^*), \leftarrow \mathbf{Eval}_{\mathsf{pp}}(F, c_1, c_2)$: As detailed above, computations on the ciphertext c_1 and the ciphertext c_2 according to the homomorphic operation F listed above output the transformed RLWE ciphertext c^*.

- $(\mathbf{m}^*) \leftarrow \mathbf{Dec}_{\mathsf{pp}}(\{\mathsf{ek}_i\}_{i\in[n]}, \{\mathsf{rk}_\ell\}_{\ell\in[n]}, c^*)$: For ciphertext $c = (a', b')$, all users calculate a partial decryption $d = a' \cdot \mathsf{uk} + e$ using their respective secret key $\mathsf{uk} \in \{\mathsf{ek}_i\}_{i\in[n]} \cup \{\mathsf{rk}_\ell\}_{\ell\in[n]}$ where $e \xleftarrow{\$} \chi'$. The decrypted output is $\mathbf{m}^* = \frac{1}{\Delta}(b' - \sum_{i=1}^{n} d_i - \sum_{\ell=1}^{n} d_\ell)$.

Lemma 1 (RLWE Multi-party Security)**.** *The RLWE multi-party instantiation achieves correctness and semantic security by Definition 3.*

Proof. We refer to [18].

3.4 Router Placing

To achieve the oblivious placing method as discussed above, we rely on [7] for their homomorphic traversal algorithm. We also include an algorithm that compiles all oblivious placings into a single array.

- $(\{c_w\}_{w\in\{0,\ldots,p\cdot n\}}) \leftarrow \mathbf{HomPlacing}(s, c, A^*)$: **HomPlacing** takes a key s, ciphertext c, an array A^* of $p\cdot n$ number of encrypted binary digits. Let a denote the integer that the bits of A^* represent. It outputs $p \cdot n$ messages such that for all $w \in \{0, \ldots, p \cdot n\} \setminus \{a\}$, c_w^* is the encrypted form of 0 and $c_a^* \coloneqq c$.

- $(\{c_w\}_{w\in\{0,\ldots,p\cdot n\}}) \leftarrow \mathbf{RtePlacing}(s, C, A)$: Given a key s, C is an array of length n contained encrypted messages from each user. Similarly, A is the encrypted indices sent by a user. **RtePlacing** returns an array of length n such that each ciphertext in C is placed into it according to the corresponding indices in A.

Our implementation of these algorithms are found in Algorithm 1 and Algorithm 2 which includes using the ciphertexts as listed in Section 3.2. We write operations F with the function **Eval** in standard notation. For example, $c_1 \cdot c_2$ is equivalent to $\mathbf{Eval}_{\mathsf{pp}}(F^{\times}, c_1, c_2)$ for ciphertexts c_1, c_2.

To give a brief explanation of Algorithm 1, it works by constructing an empty binary tree with $p \cdot n$ layers with the input message as the root. Then, for layer $w \in [p \cdot n]$, it distributes the root value to the next layer according to the input $A_w^* = \mathsf{RGSW}_s(m)$ for some key s and binary value $m \in \{0, 1\}$. The input value c, an RLWE cipherterxt, will eventually be placed at the index corresponding to the integer representation of A in the bottom layer of the tree.

Algorithm 1 Homomorphic Placing

Input: For key s, $c \coloneqq \mathsf{RLWE}_s(m)$ for message m and $A = \{\mathsf{RGSW}_s(m_w)\}_{w \in \{0,\ldots,\log(p \cdot n)-1\}}$ where $m_w \in \{0, 1\}$ and the value that A represents is a.
Output: $\{c_w^*\}_{w \in \{0,\ldots,\log(p \cdot n)-1\}}$ where $\forall w \in \{0, \ldots, p \cdot n\} \setminus \{a\} : c_w^* \coloneqq \mathsf{RLWE}_s(0)$ and $c_a^* \coloneqq c$.

function HomPlacing(s, c, A^*)
  $b_0 \coloneqq c$
  $b_x \coloneqq \mathsf{RLWE}_s(0)$ for $x \in \{1, \ldots, 2p \cdot n - 2\}$
  for $w \leftarrow \{0, \ldots, \log(p \cdot n) - 1\}$ **do**
    for $z \leftarrow \{0, \ldots, 2^w - 1\}$ **do**
      $b_{2^{w+1}+2 \cdot z-1} \coloneqq b_{2^w+z-1} - (b_{2^w+z} \cdot A_w^*)$
      $b_{2^{w+1}+2 \cdot z+1} \coloneqq (b_{2^w+z-1} \cdot A_w^*)$
    end for
  end for
  return $b_{p \cdot n}, \ldots b_{2p \cdot n-1}$
end function

Algorithm 2 Router Placing

Input: For key s, $C \coloneqq \{\mathsf{RLWE}_s(m_i)\}_{i \in [n]}$ and $A = \{A_i^*\}_{i \in [n]}$ as described in Algorithm 1.
Output: An array of length $p \cdot n$ such that each ciphertext in C is placed according to A.

function RtePlacing(s, C, A)
  $D_x \coloneqq \mathsf{RLWE}_s(0)$ for $x \in \{0, \ldots, p \cdot n - 1\}$
  for $i \leftarrow \{0, \ldots, p \cdot n - 1\}$ **do**
    $C^* \coloneqq$ HomPlacing(s, C_i, A_i)
    for $x \leftarrow \{0, \ldots, p \cdot n - 1\}$ **do**
      $D_x \coloneqq D_x + C_x^*$
    end for
  end for
  return D
end function

Note on failure In the context of our system, there is a small chance of a router failing to write a message depending on the value of p. However, the message is overwritten by noise and a user can simply send the message again, so we do not consider this. It is also

possible to increase the size bins that a router holds in order to decrease the failure rate. We point to [22] for a more concrete analysis for failure rate, though we do include experimental results in Table 4.

4 ARN Construction

4.1 ARN with Sender Anonymity

Let $\mathsf{MP} \coloneqq (\mathbf{Setup}, \mathbf{Enc}, \mathbf{Eval}, \mathbf{Dec})$ denote a multi-party fully homomorphic encryption scheme. Specifically, we can instantiate the MP scheme with the **RLWE** scheme in Section 3.3. Let B denote the uniform distribution of arrays of length $\log(p \cdot n)$ of which all elements are either 0 or 1.

$\mathbf{ARN}_{\mathsf{SA}}$

- $\mathbf{Setup}(1^\lambda, n, k)$: Output $\mathsf{MP}.\mathbf{Setup}(1^\lambda, n, k)$.
- $\mathbf{Enc}(\widehat{\mathsf{pk}}, m_i, \pi_i)$:
  - Let $u \xleftarrow{\$} [k]$ and c'_j be the encrypted message by the ith sender for the the jth router such that

$$c'_j \coloneqq \begin{cases} \mathsf{MP}.\mathbf{Enc}(\widehat{\mathsf{pk}}, m_i) & \text{if } j = u, \\ \mathsf{MP}.\mathbf{Enc}(\widehat{\mathsf{pk}}, 0) & \text{otherwise.} \end{cases}$$

  - Sample $a_i \xleftarrow{\$} B$.
  - Let A_i be the encrypted form[a] of a_i such that $(A_i)_x \leftarrow \mathsf{MP}.\mathbf{Enc}(\widehat{\mathsf{pk}}, (a_{i,u})_x)$ where $x \in [p \cdot n]$ is the index of a value within an array.
  - Output $\{c_{i,j}\}_{j \in [k]}$ such that $c_{i,j} \coloneqq (c'_{i,j}, A_i)$
- $\mathbf{Rte}(\widehat{\mathsf{pk}}, \{c_{i,j}\}_{i \in [n]})$:
  - Parse $c_{i,j} \coloneqq (c'_{i,j}, A_i)$
  - Let $C \coloneqq \{c'_{i,j}\}_{i \in [n]}$ and $A \coloneqq \{A_i\}_{i \in [n]}$
  - Return $\mathbf{RtePlacing}(\widehat{\mathsf{pk}}, C, A)$.
- $\mathbf{Dec}(\{\mathsf{ek}_i\}_{i \in [n]}, \{\mathsf{rk}_\ell\}_{\ell \in [n]}, C_j)$: Output $\mathcal{M}_j$ such that $\forall m^*_j \in \mathcal{M}_j$, $\mathbf{m}^*_j \leftarrow \mathsf{MP}.\mathbf{Dec}(\{\mathsf{ek}_i\}_{i \in [n]}, \{\mathsf{rk}_\ell\}_{\ell \in [n]}, c^*_j)$ where $c^*_j \in C_j$ and $m^*_j \neq 0$.

[a] A_i takes the form of $\mathsf{RGSW}_{\widehat{\mathsf{pk}}}(a_i)$ as described in Section 3.2.

Theorem 5 (ARN with sender anonymity)**.** *Suppose that the underlying FHE scheme satisfies semantic security by Definition 3. Then,* $\boldsymbol{ARN}_{\mathsf{SA}}$ *achieves sender anonymity by Definition 1.*

Proof. We show that any admissible p.p.t. adversary's view of $\mathsf{ARN\text{-}Expt}^{0,\mathcal{A}}(1^\lambda)$ and $\mathsf{ARN\text{-}Expt}^{1,\mathcal{A}}(1^\lambda)$ are computationally indistinguishable through a sequence of hybrid experiments. We give a direct reduction between the underlying FHE scheme and hybrid experiments.

Experiments Hyb_i **for** $i \in \{0, \ldots, n\}$**.** If $i = 0$, the experiment Hyb_0 is defined identically to $\mathsf{ARN\text{-}Expt}^{0,\mathcal{A}}(1^\lambda)$. Otherwise, the experiment Hyb_i is defined identically to Hyb_{i-1} except that the i-th sender and the sender that sends $m_i \in \mathcal{M}_1$ swap messages. In other words, sender i swaps messages with whichever sender in Hyb_{i-1} is sending m_i when $b = 1$. If the i-th sender is already sending $m_i \in \mathcal{M}_1$, then Hyb_i is identical to Hyb_{i-1}. We note that that Hyb_n is identical to $\mathsf{ARN\text{-}Expt}^{1,\mathcal{A}}(1^\lambda)$ by the nature of its construction.

Claim 1. *Suppose that the underlying FHE scheme satisfies semantic security by Definition 3. Then, for any* $i \in [n]$*, any p.p.t. adversary's views in* Hyb_i *and* Hyb_{i-1} *are computationally indistinguishable.*

Proof. Due to the admissibility rules that $\mathcal{A}$ must abide by, the multisets of messages in both hybrids, $\mathcal{M}_i, \mathcal{M}_{i-1}$ must remain the same. Thus, it suffices to show that if a p.p.t. adversary $\mathcal{A}$ that respects the ARN experiment's admissibility rules can distinguish Hyb_i and Hyb_{i-1} with non-negligible probability, we can build a reduction $\mathcal{A}_{\mathsf{FHE}}$ with knowledge of an encryption oracle $\mathcal{O}$ that breaks the IND-CPA security of the underlying FHE scheme with a challenger Ch.

It is sufficient to use an similar version of the standard multi-message IND-CPA experiment to define the underlying FHE scheme. We formalize both versions of multi-message security and defer the proof of the reduction to the standard experiment and additional details to Appendix B.

1. First, $\mathcal{A}$ sends $n, k, \mathcal{K}_S, \mathcal{K}_R$ to $\mathcal{A}_{\mathsf{FHE}}$ in which $\mathcal{A}_{\mathsf{FHE}}$ sends $|\mathcal{K}_S| + |\mathcal{K}_R|$, $2n$, and k to Ch.
2. Then, Ch returns $\widehat{\mathsf{pk}}$ and $|\mathcal{K}_S| + |\mathcal{K}_R|$ number of secret keys to $\mathcal{A}_{\mathsf{FHE}}$, which are forwarded to $\mathcal{A}$.
3. During each round $t \in \mathbb{N}$, in which $\mathcal{A}_{\mathsf{FHE}}$ returns the output of $\mathcal{A}$, the following is performed:
   (a) $\mathcal{A}$ sends $\{\{(i, \pi_i^b, m_i^b)\}_{i\in[n]}\}_{b\in\{i,i-1\}}$ to $\mathcal{A}_{\mathsf{FHE}}$ in which $\mathcal{A}_{\mathsf{FHE}}$ forwards the messages of the two differing tuples and the routers that will transmit said messages to

Ch. We denote these as (m_x, m_y) and $(u_x \in [k], u_y \in [k])$ respectively. If there are no differing tuples, $\mathcal{A}_{\mathsf{FHE}}$ forwards any two messages from distinct tuples.

(b) Ch samples $b \leftarrow\!\!\$\ \{i, i-1\}$ and if $b = i$, sends back ciphertexts $(\vec{c}_x, \vec{c}_y)$ where

$$\vec{c}_x \coloneqq \{\mathsf{MP}.\mathbf{Enc}(0)_1, \ldots, \mathsf{MP}.\mathbf{Enc}(m_x)_{u_x}, \ldots, \mathsf{MP}.\mathbf{Enc}(0)_k\},$$

$$\vec{c}_y \coloneqq \{\mathsf{MP}.\mathbf{Enc}(0)_1, \ldots, \mathsf{MP}.\mathbf{Enc}(m_y)_{u_y}, \ldots, \mathsf{MP}.\mathbf{Enc}(0)_k\};$$

else if $b = 1$ sends $(\vec{c}_x, \vec{c}_y)$ where

$$\vec{c}_x \coloneqq \{\mathsf{MP}.\mathbf{Enc}(0)_1, \ldots, \mathsf{MP}.\mathbf{Enc}(m_y)_{u_x}, \ldots, \mathsf{MP}.\mathbf{Enc}(0)_k\},$$

$$\vec{c}_y \coloneqq \{\mathsf{MP}.\mathbf{Enc}(0)_1, \ldots, \mathsf{MP}.\mathbf{Enc}(m_x)_{u_y}, \ldots, \mathsf{MP}.\mathbf{Enc}(0)_k\}.$$

Along with the received messages, $\mathcal{A}_{\mathsf{FHE}}$ sends the result of $\mathbf{Enc}$ using $\mathcal{O}$ for $i \in \mathcal{H}_S$ to $\mathcal{A}$ including all index arrays.

(c) Then, $\mathcal{A}$ performs $C_j \leftarrow \mathbf{Rte}(\widehat{\mathsf{pk}}, \{c_i\}_{i\in[n]})$ for each router $j \in [k]$ with received messages and self chosen c_i for $i \in \mathcal{K}_S$.

(d) The messages are collectively decrypted as $\mathcal{M}_j \leftarrow \mathbf{Dec}(\{\mathsf{ek}_i\}_{i\in[n]}, \{\mathsf{rk}_\ell\}_{\ell\in[n]}, C_j)$ for all $j \in [k]$ with $\mathcal{A}_{\mathsf{FHE}}$ acting for all honest users.

We note that because the routing algorithm places homomorphically, $\mathcal{A}_{\mathsf{FHE}}$ cannot correlate the position of messages in the output of $\mathbf{Dec}$. Thus, $\mathcal{A}_{\mathsf{FHE}}$ has a negligible advantage to break the game, meaning $\mathcal{A}_{\mathsf{FHE}}$ views of Hyb_i and Hyb_{i-1} are computationally indistinguishable. □

The proof of Theorem 5 now follows due to Claim 1. □

4.2 ARN with Receiver Anonymity

Let $\mathsf{PKE} \coloneqq (\mathbf{Gen}, \mathbf{Enc}, \mathbf{Dec})$ denote some standard public key encryption scheme as detailed in Appendix A. We note that $\mathbf{Gen}$ does not output identifiable information about the user performing the algorithm.

$\mathbf{ARN}_{\mathsf{RA}}$

- **Setup**$(1^\lambda, n, k)$:
  - $\widehat{\mathsf{pk}}^*, \{\mathsf{ek}_i\}_{i\in[n]}, \{\mathsf{rk}_\ell\}_{\ell\in[n]} \coloneqq \mathbf{MP.Setup}_{\mathsf{pp}}(1^\lambda, n, k)$
  - For $\ell \in [n]$, $\mathsf{pk}_\ell, \mathsf{sk}_\ell \leftarrow \mathsf{PKE}.\mathbf{Gen}(1^\lambda)$
  - Run the $\mathbf{ARN}_{\mathsf{SA}}$ protocol with sender/receiver role reversal such that:
    - For $\ell \in [n]$, $\{c'_{\ell,j}\}_{j\in[k]} \leftarrow \mathbf{ARN}_{\mathsf{SA}}.\mathbf{Enc}(\widehat{\mathsf{pk}}^*, \mathsf{pk}_\ell, 0)$
    - For $j \in [k]$, $(C'_j) \leftarrow \mathbf{ARN}_{\mathsf{SA}}.\mathbf{Rte}(\widehat{\mathsf{pk}}, \{c'_{\ell,j}\}_{\ell\in[n]})$
    - $\mathsf{pk} \leftarrow \mathbf{ARN}_{\mathsf{SA}}.\mathbf{Dec}(\{\mathsf{ek}_i\}_{i\in[n]}, \{\mathsf{rk}_\ell\}_{\ell\in[n]}, C'_j)$ s.t. pk is the set of all PKE public keys.
  - Output $\widehat{\mathsf{pk}}, \{\mathsf{ek}_i\}_{i\in[n]}, \{\mathsf{rk}_\ell\}_{\ell\in[n]}$ where $\widehat{\mathsf{pk}} \coloneqq (\widehat{\mathsf{pk}}^*, \mathsf{pk})$
- **Enc**$(\widehat{\mathsf{pk}}, m_i, \pi_i)$: Output $\mathbf{ARN}_{\mathsf{SA}}.\mathbf{Enc}(\widehat{\mathsf{pk}}^*, c_i^*, \pi_i)$ such that $\widehat{\mathsf{pk}} \coloneqq (\widehat{\mathsf{pk}}^*, \mathsf{pk})$ and $c_i^* \coloneqq \mathsf{PKE}.\mathbf{Enc}(\mathsf{pk}_{\pi_i}, m_i)$.
- **Rte**$(\widehat{\mathsf{pk}}, \{c_{i,j}\}_{i\in[n]})$: Output $\mathbf{ARN}_{\mathsf{SA}}.\mathbf{Rte}(\widehat{\mathsf{pk}}, \{c_{i,j}\}_{i\in[n]})$.
- **Dec**$(\{\mathsf{ek}_i\}_{i\in[n]}, \{\mathsf{rk}_\ell\}_{\ell\in[n]}, C_j)$: Output $\mathbf{ARN}_{\mathsf{SA}}.\mathbf{Dec}(\{\mathsf{ek}_i\}_{i\in[n]}, \{\mathsf{rk}_\ell\}_{\ell\in[n]}, C_j)$.

Theorem 6 (ARN with receiver anonymity)**.** *Suppose that the PKE scheme satisfies semantic security as defined in Appendix A. Then, $\boldsymbol{ARN}_{RA}$ achieves receiver anonymity by Definition 2.*

Proof Given Theorem 5, the ciphertexts shuffled in the **Rte** step cannot be tied to a specific honest sender. Thus, it suffices to prove Theorem 6 through the same sequence of hybrids as stated in Section 4.1 in which we use the $\mathbf{ARN}_{\mathsf{RA}}$ scheme instead of $\mathbf{ARN}_{\mathsf{SA}}$.

Claim 2. *Suppose that the underlying PKE scheme satisfies semantic security by Definition 3. Then, for any $i \in [n]$, any p.p.t. adversary's views in Hyb_i and Hyb_{i-1} are computationally indistinguishable.*

Proof We note that the only difference between this and the previous section is that messages are encrypted with recipient π_i's public key first. Thus, due to Claim 1, a straightforward reduction can be made from the standard multi-message scheme as detailed in Appendix B to PKE. □

Since the adversary cannot identity an honest recipient by their PKE public key given Theorem 5, the proof of Theorem 6 now follows due to Claim 2. □

5 Implementation

We assume that all participants agree on the FHE parameters as detailed in Table 3 before the protocol begins, including the needed variables for the TFHE variants [6] of RLWE which we can employ. This yields 115 bits of security according to the Lattice-Estimator[1] library [1]. We note that to aptly contrast our results with sPAR, we use similar parameters.

For implementation, we use the core-crypto module from the TFHE-rs[2] library [23] in the Rust programming language. All algorithms were benchmarked using a dedicated computer running Arch Linux with an Intel Core i5-14400F Raptor Lake 10-Core and 32GB of RAM.

polynomial degree N	2048
error standard deviation	2^{-55}
plaintext modulus t	2^8
ciphertext modulus q	2^{64}
RGSW encryption (β, α)	$(2^5, 9)$

Table 3: Implementation variables.

5.1 Benchmarks

We simulate the success rates of total message transmission and benchmark ARN with sender anonymity for a different number of clients ($n \in \{32, 64, 128\}$) with differing number of routers ($k \in \{1, 2, 4, 8, 16, 32\}$). In terms of the bin multiplier p, we show two different types of values, constant ($p = 16$) and variable ($p = 32/k$), for both the simulation and benchmarks.

We see in Table 4 that while having a constant value p, transmission success dramatically increases. However, this results in more computation time given that there are an increasing amount of bins with respect to k. Thus, we scale the amount of total bins with k in which the transmission success remains relatively constant.

n	k	p	success rate	p	success rate
32, 64, 128	1	16	0.941 ± 0.004	32/k	0.971 ± 0.002
32, 64, 128	2	16	0.968 ± 0.003	32/k	0.966 ± 0.002
32, 64, 128	4	16	0.985 ± 0.001	32/k	0.969 ± 0.002
32, 64, 128	8	16	0.992 ± 0.002	32/k	0.967 ± 0.003
32, 64, 128	16	16	0.996 ± 0.001	32/k	0.970 ± 0.002
32, 64, 128	32	16	$0.998 \pm (< 0.000)$	32/k	0.973 ± 0.004

Table 4: Message success simulation (average of 100).

[1] Commit #a51a410.
[2] Version 1.5.4.

With respect to adding routers for a constant bin multiplier p, Table 5 shows that computation time does not significantly change. There is a slight increase in encryption times when adding more routers; however, the change is trivial in comparison to routing protocol. We include a more detailed showcase of the dataset in Appendix C.

n	k	**Setup**	**Enc** per message	**Rte** per message	**Dec** per n messages
32	$\{1, 2, 4, 8, 16, 32\}$	730 ± 19	2122 ± 687	7653 ± 121	714 ± 24
64	$\{1, 2, 4, 8, 16, 32\}$	1430 ± 8	2279 ± 680	15069 ± 150	1412 ± 29
128	$\{1, 2, 4, 8, 16, 32\}$	2894 ± 45	2468 ± 680	29956 ± 334	2838 ± 78

Table 5: Computation time in milliseconds for $p = 16$ (average of 10).

However, when adding routers with p scaled by the number of routers k, the time needed for the routing algorithm significantly decreases as seen in Table 6. While there is a noticeable increase in encryption times as k increases, it is eclipsed by the reduction in routing time.

n	k	**Setup**	**Enc** per message	**Rte** per message	**Dec** per n messages
32	1	726	1770	16554	685
32	2	705	1629	8337	691
32	4	705	1544	4219	687
32	8	707	1549	2179	686
32	16	705	1733	1142	687
32	32	707	2283	613	693
64	1	1426	1922	32892	1386
64	2	1425	1815	16552	1386
64	4	1426	1732	8281	1373
64	8	1411	1715	4210	1373
64	16	1410	1902	2169	1373
64	32	1410	2434	1134	1374
128	1	2819	2096	65149	2748
128	2	2825	1971	32582	2751
128	4	2824	1887	16379	2750
128	8	2820	1891	8278	2748
128	16	2823	2069	4214	3088
128	32	3228	2609	2173	2815

Table 6: Computation time in milliseconds for $p = 32/k$ (average of 10).

6 Discussion and Open Questions

While this communication system may not be able to scale to the size of other anonymous communication systems, it is a step in that direction. More specifically, we consider our work as a meaningful extension to the work of Das and Park [8].

We do not directly compare speeds of ARN to sPAR; however, we can draw some comparisons. The core of the scheme is similar in construction outside of the placing algorithm, in which sPAR has a more efficient version. Thus, if ARN included the placing algorithm from sPAR, we theorize that computation time would be drastically cut.

Outside of performance, there is a real-world scenario where a router is unable to continue operating. Given that routers act independently, our system continues to send a subset of messages even if some routers stop sending.

Additionally, our construction provokes some thoughts in a similar vein to the typical threshold model. Namely, routers and users spend computational time on meaningless messages to achieve anonymity. To minimize this, users could pick a subset of routers to send messages, which in turn would lessen computation time while sacrificing security. This has a theoretical benefit of only certain users being included in the interactive decryption phase.

Given this step in constructing a scalable system and this proposed direction in threshold security, some questions naturally arise:

1. What are the concrete performance benefits of combining the placing algorithm from sPAR with the addition of more routers?

2. What are the security implications when users send messages to a subset of routers?

References

[1] Martin R. Albrecht, Rachel Player, and Sam Scott. "On the concrete hardness of Learning with Errors". In: *Journal of Mathematical Cryptology* 9 (2015), pp. 169–203. URL: https://api.semanticscholar.org/CorpusID:86408.

[2] Gilad Asharov et al. "Multiparty Computation with Low Communication, Computation and Interaction via Threshold FHE". In: *Advances in Cryptology – EUROCRYPT 2012*. Ed. by David Pointcheval and Thomas Johansson. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 483–501. ISBN: 978-3-642-29011-4.

[3] Michael Backes et al. "AnoA: A Framework for Analyzing Anonymous Communication Protocols". In: *2013 IEEE 26th Computer Security Foundations Symposium*. 2013, pp. 163–178. DOI: 10.1109/CSF.2013.18.

[4] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. "(Leveled) Fully Homomorphic Encryption without Bootstrapping". In: *ACM Trans. Comput. Theory* 6.3 (July 2014). ISSN: 1942-3454. DOI: 10.1145/2633600. URL: https://doi.org/10.1145/2633600.

[5] Zvika Brakerski and Vinod Vaikuntanathan. "Fully Homomorphic Encryption from Ring-LWE and Security for Key Dependent Messages". In: *Advances in Cryptology – CRYPTO 2011*. Ed. by Phillip Rogaway. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 505–524. ISBN: 978-3-642-22792-9.

[6] Ilaria Chillotti et al. "TFHE: Fast Fully Homomorphic Encryption Over the Torus". In: 33.1 (Jan. 2020), pp. 34–91. ISSN: 0933-2790. DOI: 10.1007/s00145-019-09319-x. URL: https://doi.org/10.1007/s00145-019-09319-x.

[7] Kelong Cong et al. "SortingHat: Efficient Private Decision Tree Evaluation via Homomorphic Encryption and Transciphering". In: *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*. CCS '22. Los Angeles, CA, USA: Association for Computing Machinery, 2022, pp. 563–577. ISBN: 9781450394505. DOI: 10.1145/3548606.3560702. URL: https://doi.org/10.1145/3548606.3560702.

[8] Debajyoti Das and Jeongeun Park. *sPAR: (Somewhat) Practical Anonymous Router*. Cryptology ePrint Archive, Paper 2025/860. 2025. URL: https://eprint.iacr.org/2025/860.

[9] Roger Dingledine, Nick Mathewson, and Paul Syverson. "Tor: The Second-Generation Onion Router". In: *13th USENIX Security Symposium (USENIX Security 04)*. San Diego, CA: USENIX Association, Aug. 2004. URL: https://www.usenix.org/conference/13th-usenix-security-symposium/tor-second-generation-onion-router.

[10] Kunjie Ge and Yongzhong He. "Detection of Sybil Attack on Tor Resource Distribution". In: *2020 IEEE International Conference on Power, Intelligent Computing and Systems (ICPICS)*. 2020, pp. 328–332. DOI: 10.1109/ICPICS50287.2020.9202013.

[11] Craig Gentry. “A fully homomorphic encryption scheme”. In: 2009. URL: https://api.semanticscholar.org/CorpusID:53903759.

[12] Craig Gentry, Amit Sahai, and Brent Waters. “Homomorphic Encryption from Learning with Errors: Conceptually-Simpler, Asymptotically-Faster, Attribute-Based”. In: *Advances in Cryptology – CRYPTO 2013*. Ed. by Ran Canetti and Juan A. Garay. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 75–92. ISBN: 978-3-642-40041-4.

[13] Robin Jadoul, Barry van Leeuwen, and Oliver Zajonc. *Multiparty FHE Redefined: A Framework for Unlimited Participants*. Cryptology ePrint Archive, Paper 2025/965. 2025. URL: https://eprint.iacr.org/2025/965.

[14] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. “On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption”. In: *STOC ’12 - Proceedings of the 2012 ACM Symposium on Theory of Computing*. Proceedings of the Annual ACM Symposium on Theory of Computing. 44th Annual ACM Symposium on Theory of Computing, STOC ’12 ; Conference date: 19-05-2012 Through 22-05-2012. 2012, pp. 1219–1234. ISBN: 9781450312455. DOI: 10.1145/2213977.2214086.

[15] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. “On ideal lattices and learning with errors over rings”. In: EUROCRYPT’10. French Riviera, France: Springer-Verlag, 2010, pp. 1–23. ISBN: 3642131891. DOI: 10.1007/978-3-642-13190-5_1. URL: https://doi.org/10.1007/978-3-642-13190-5_1.

[16] C.A. Melchor and Y. Deswarte. “From DC-Nets to pMIXes: Multiple Variants for Anonymous Communications”. In: *Fifth IEEE International Symposium on Network Computing and Applications (NCA’06)*. 2006, pp. 163–172. DOI: 10.1109/NCA.2006.32.

[17] Christian Mouchet et al. *Multiparty Homomorphic Encryption from Ring-Learning-With-Errors*. Cryptology ePrint Archive, Paper 2020/304. 2020. DOI: 10.2478/popets-2021-0071. URL: https://eprint.iacr.org/2020/304.

[18] Christian Mouchet et al. “Multiparty Homomorphic Encryption from Ring-Learning-with-Errors”. In: *Proceedings on Privacy Enhancing Technologies* 2021 (Oct. 2021), pp. 291–311. DOI: 10.2478/popets-2021-0071.

[19] Jeongeun Park. “Homomorphic Encryption for Multiple Users With Less Communications”. In: *IEEE Access* 9 (2021), pp. 135915–135926. DOI: 10.1109/ACCESS.2021.3117029.

[20] Krishna Sampigethaya and Radha Poovendran. “A Survey on Mix Networks and Their Secure Applications”. In: *Proceedings of the IEEE* 94.12 (2006), pp. 2142–2181. DOI: 10.1109/JPROC.2006.889687.

[21] Elaine Shi and Ke Wu. *Non-Interactive Anonymous Router*. Cryptology ePrint Archive, Paper 2021/435. 2021. URL: https://eprint.iacr.org/2021/435.

[22] David Wagner. “A Generalized Birthday Problem”. In: *Proceedings of the 22nd Annual International Cryptology Conference on Advances in Cryptology*. CRYPTO ’02. Berlin, Heidelberg: Springer-Verlag, 2002, pp. 288–303. ISBN: 354044050X.

[23] Zama. *TFHE-rs: A Pure Rust Implementation of the TFHE Scheme for Boolean and Integer Arithmetics Over Encrypted Data*. `https://github.com/zama-ai/tfhe-rs`. 2022.

A Public Key Encryption

A public key encrytion scheme $\mathsf{PKE} \coloneqq (\mathbf{Gen}, \mathbf{Enc}, \mathbf{Dec})$ has three randomized algorithms:

1. $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathbf{Gen}(1^\lambda)$ generates a secret and public key,
2. $c \leftarrow \mathbf{Enc}(\mathsf{pk}, m)$ encrypts a plaintext message $m \in \{0,1\}^\ell$ with the public key where $\ell(\cdot)$ is a fixed polynomial function in λ, and
3. $m \leftarrow \mathbf{Dec}(c, \mathsf{sk})$ decrypts a ciphertext with the secret key.

Correctness requires that for any λ and $m \in \{0,1\}^\ell$, with probability 1, $\mathsf{pk}, \mathsf{sk} \leftarrow \mathbf{Gen}(1^\lambda)$ and $c \leftarrow \mathbf{Enc}(\mathsf{pk}, m)$, $\mathbf{Dec}(c, \mathsf{sk})$ must output m.

Semantic security requires that as long as the non-uniform p.p.t. adversary knows the honestly generated secret key, it cannot distinguish two honestly generated ciphertexts of arbitrary messages m_1, m_2. More precisely, there exists a p.p.t. adversary $\mathcal{A}$, such that the following experiment is computationally indistinguishable with $b \in \{0,1\}$:

- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathbf{Gen}(1^\lambda)$.
- $(m_0, m_1) \leftarrow \mathcal{A}(1^\lambda, \widehat{\mathsf{pk}})$.
- $c \leftarrow \mathbf{Enc}(\mathsf{pk}, m_b)$.

B Reduction to Standard Multi-Message IND-CPA Security

We prove the reduction from the standard multi-message IND-CPA experiment to our modified IND-CPA experiment. Let $\mathsf{MP} \coloneqq (\mathbf{Setup}, \mathbf{Enc}, \mathbf{Eval}, \mathbf{Dec})$ denote a multi-party fully homomorphic encryption scheme as detailed in Section 3.1. We note that, per the security of the MPscheme, an adversary must not corrupt all parties.

Let the standard experiment be defined as the following where the oracle $\mathcal{O}^b_{\mathsf{pk}}$ takes queries of messages pairs of the same length (m_0, m_1) and returns $\mathsf{MP}.\mathbf{Enc}(\mathsf{pk}, m_b)$:

$\mathsf{STD\text{-}Expt}^{b,\mathcal{A}}(1^\lambda)$

- $(|\mathcal{K}|, n, k) \leftarrow \mathcal{A}(1^\lambda)$.
- $(\mathsf{pk}, \{\mathsf{sk}\}_{i\in[2n]}) \leftarrow \mathsf{MP}.\mathbf{Setup}(1^\lambda, 2n, k)$.
- $b' \leftarrow \mathcal{A}^{\mathcal{O}^b_{\mathsf{pk}}}(1^\lambda, \{\mathsf{sk}_i\}_{i\in\mathcal{K}})$.

In our modified experiment, the oracle $\widetilde{\mathcal{O}}^b_{\mathsf{pk}}$ takes queries of messages pairs (m_0, m_1) of the same along with a pair of integers (u_0, u_1) such that $u_0, u_1 \in [k]$. The oracle returns a vector of ciphertexts such that if $b = 0$,

$$\vec{c}_x \coloneqq \{\mathsf{MP}.\mathbf{Enc}(0)_1, \ldots, \mathsf{MP}.\mathbf{Enc}(m_0)_{u_0}, \ldots, \mathsf{MP}.\mathbf{Enc}(0)_k\},$$

$$\vec{c}_y \coloneqq \{\mathsf{MP}.\mathbf{Enc}(0)_1, \ldots, \mathsf{MP}.\mathbf{Enc}(m_1)_{u_1}, \ldots, \mathsf{MP}.\mathbf{Enc}(0)_k\};$$

else if $b = 1$,

$$\vec{c}_x \coloneqq \{\mathsf{MP}.\mathbf{Enc}(0)_1, \ldots, \mathsf{MP}.\mathbf{Enc}(m_1)_{u_0}, \ldots, \mathsf{MP}.\mathbf{Enc}(0)_k\},$$

$$\vec{c}_y \coloneqq \{\mathsf{MP}.\mathbf{Enc}(0)_1, \ldots, \mathsf{MP}.\mathbf{Enc}(m_0)_{u_1}, \ldots, \mathsf{MP}.\mathbf{Enc}(0)_k\}.$$

Let the modified experiment be defined as the following:

$\mathsf{MOD\text{-}Expt}^{b,\mathcal{A}}$

- $|\mathcal{K}|, n, k \leftarrow \mathcal{A}(1^\lambda)$
- $\mathsf{pk}, \{\mathsf{sk}_i\}_{i\in[2n]} \leftarrow \mathsf{MP}.\mathbf{Setup}(2n, k)$
- $b' \leftarrow \mathcal{A}^{\widetilde{\mathcal{O}}^b_{\mathsf{pk}}}(1^\lambda, \{\mathsf{sk}_i\}_{i\in\mathcal{K}})$.

Lemma 2 (Multi-message IND-CPA reduction)**.** *If a p.p.t. adversary can distinguish* $\mathsf{MOD\text{-}Expt}^b$ *with non-negligible probability, then it can distinguish* $\mathsf{STD\text{-}Expt}^b$ *with non-negligible probability.*

Proof It is sufficient to show a direct reduction between the two experiments. Specifically, we show that if any p.p.t. adversary $\mathcal{A}$ with access to the oracle $\mathcal{O}$ can distinguish $\mathsf{MOD\text{-}Expt}^0$ and $\mathsf{MOD\text{-}Expt}^1$, then $\mathcal{A}$ can also distinguish $\mathsf{STD\text{-}Expt}^0$ and $\mathsf{STD\text{-}Expt}^1$. With Ch as the challenger, $\mathcal{A}_{\mathsf{MOD}}$ as the modified experiment adversary, and $\mathcal{A}_{\mathsf{STD}}$ as the standard experiment adversary, the reduction is as follows.

1. First, $\mathcal{A}_{\mathsf{MOD}}$ sends $|\mathcal{K}|, n, k$ to $\mathcal{A}_{\mathsf{STD}}$ which is forwarded to Ch.
2. Then, $\widehat{\mathsf{pk}}, \{\mathsf{sk}_u\}_{u\in[\mathcal{K}]}$ is returned by Ch to $\mathcal{A}_{\mathsf{STD}}$ which is forwarded to $\mathcal{A}_{\mathsf{MOD}}$.
3. For a polynomial number of rounds,
   (a) $\mathcal{A}_{\mathsf{MOD}}$ sends $m_0,\ m_1,\ u_0,\ u_1$ to $\mathcal{A}_{\mathsf{STD}}$.
   (b) $\mathcal{A}_{\mathsf{STD}}$ forwards a sequence of message pairs $\{(0,0)_1, \ldots (m_0, m_1)_{u_0}, \ldots, (0,0)_k\}$ individually to Ch, to which Ch responds with $\vec{c}_x = \{(c)_1, \ldots (c_b)_{u_0}, \ldots, (c)_k\}$. We note that each message gets a response before $\mathcal{A}_{\mathsf{STD}}$ sends the next message.
   (c) Similarly, $\mathcal{A}_{\mathsf{STD}}$ sends the same messages except using u_1 instead of u_0. In response, Ch returns the sequence $\vec{c}_y = \{(c)_1, \ldots (c_{b-1})_{u_1}, \ldots, (c)_k\}$.
   (d) $\mathcal{A}_{\mathsf{STD}}$ forwards $(\vec{c}_x, \vec{c}_y)$ to $\mathcal{A}_{\mathsf{MOD}}$.

Thus, $\mathcal{A}_{\mathsf{MOD}}$ and $\mathcal{A}_{\mathsf{STD}}$ views of the experiments when $b = 0$ or $b = 1$ are the same. The proof of Lemma 2 now follows due to the reduction. □

C Full Benchmarks

We include the full dataset from the summarized Table 5. It shows that while benchmarks are somewhat stagnant independent of the number of routers, there is an increase of computational time in the **Enc** algorithm. Additionally, there is a noticeable decrease in time of the **Rte** algorithm as the number of routers increase.

n	k	**Setup**	**Enc** per message	**Rte** per message	**Dec** per n messages
32	1	717	1435	7775	720
32	2	711	1483	7643	702
32	4	717	1565	7612	697
32	8	750	1796	7633	697
32	16	718	2159	7661	739
32	32	711	2810	7532	690
64	1	1427	1599	15059	1398
64	2	1439	1662	15220	1410
64	4	1439	1740	15058	1395
64	8	1424	1901	15005	1383
64	16	1437	2289	15112	1442
64	32	1422	2959	14919	1383
128	1	2885	1788	30290	2820
128	2	2927	1849	29927	2763
128	4	2870	1903	29622	2761
128	8	2854	2093	29792	2916
128	16	2940	2504	30096	2796
128	32	2849	3148	29810	2760

Table 7: Computation time for $p = 16$ in milliseconds (average of 10).