Practical and Provably Secure Onion Routing

Megumi Ando
Computer Science Department, Brown University, Providence, RI 02912 USA
mando@cs.brown.edu

Anna Lysyanskaya
Computer Science Department, Brown University, Providence, RI 02912 USA
anna@cs.brown.edu

Eli Upfal
Computer Science Department, Brown University, Providence, RI 02912 USA
eli@cs.brown.edu

Abstract

In an onion routing protocol, messages travel through several intermediaries before arriving at their destinations; they are wrapped in layers of encryption (hence they are called "onions"). The goal is to make it hard to establish who sent the message. It is a practical and widespread tool for creating anonymous channels.

For the standard adversary models – passive and active – we present practical and provably secure onion routing protocols. Akin to Tor, in our protocols each party independently chooses the routing paths for his onions. For security parameter λ, our differentially private solution for the active adversary takes $O(\log^2 \lambda)$ rounds and requires every participant to transmit $O(\log^4 \lambda)$ onions in every round.

2012 ACM Subject Classification Security and privacy → Security protocols

Keywords and phrases Anonymity, traffic analysis, statistical privacy, differential privacy

Digital Object Identifier 10.4230/LIPIcs.ICALP.2018.144

Related Version A full version of the paper is available at `https://arxiv.org/abs/1706.05367`.

Funding This work was supported in part by NSF grants IIS-1247581 and CNS-1422361.

Acknowledgements We'd like to thank the reviewers for their thoughtful comments.

1 Introduction

Anonymous channels are a prerequisite for protecting user privacy. But how do we achieve anonymous channels in an Internet-like network that consists of point-to-point links?

If a user Alice wishes to send a message m to a user Bob, she may begin by encrypting her message m under Bob's public key to obtain the ciphertext $c_{Bob} = \mathsf{Enc}(\mathsf{pk}_{Bob}, m)$. But sending c_{Bob} directly to Bob would allow an eavesdropper to observe that Alice is in communication with Bob. So instead, Alice may designate several intermediate relays, called "mix-nodes" (typically chosen at random) and send the ciphertext through them, "wrapped" in several layers of encryption so that the ciphertext received by a mix-node cannot be linked to the ciphertext sent out by the mix-node. Each node decrypts each ciphertext it receives ("peels off" a layer of encryption) and discovers the identity of the next node and the ciphertext to send along. This approach to hiding who is talking to whom is called "onion routing" [10] (sometimes it is also called "anonymous remailer" [13]) because the ciphertexts are layered, akin to onions; from now on we will refer to such ciphertexts as "onions".

Onion routing is attractive for several reasons: (1) simplicity: users and developers understand how it works; the only cryptographic tool it uses is encryption; (2) fault-tolerance: it can easily tolerate and adapt to the failure of a subset of mix-nodes; (3) scalability: its performance remains the same even as more and more users and mix-nodes are added to the system. As a result, onion routing is what people use to obscure their online activities. According to current statistics published by the Tor Project, Inc., Tor is used by millions of users every day to add privacy to their communications [14, 15][1].

In spite of its attractiveness and widespread use, the security of onion routing is not well-understood.

The definitional question – what notion of security do we want to achieve? – has been studied [3, 20, 21, 28]. The most desirable notion, which we will refer to as "statistical privacy", requires that the adversary's view in the protocol be distributed statistically independently of who is trying to send messages to whom[2]. Unfortunately, a network adversary observing the traffic flowing out of Alice and flowing into Bob can already make inferences about whether Alice is talking to Bob. For example, if the adversary knows that Alice is sending a movie to someone, but there isn't enough traffic flowing into Bob's computer to suggest that Bob is receiving a movie, then Bob cannot be Alice's interlocutor. (Participants' inputs may also affect others' privacy in other ways [20].)

So let us consider the setting in which, in principle, statistical privacy can be achieved: every party wants to anonymously send and receive just one short message to and from some other party. Let us call this "the simple input-output (I/O) setting". In the simple I/O setting, anonymity can be achieved even against an adversary who can observe the entire network if there is a trusted party through whom all messages are routed. Can onion routing that does not rely on one trusted party emulate such a trusted party in the presence of a powerful adversary?

Specifically, we may be dealing with *the network adversary* that observes all network traffic; or the stronger *passive adversary* that, in addition to observing network traffic, also observes the internal states of a fraction of the network nodes; or the most realistic *active adversary* that observes network traffic and also controls a fraction of the nodes. Prior work analyzing Tor [3, 20, 21] did not consider these standard adversary models. Instead, they focused on the adversary who was entirely absent from some regions of the network, but resourceful adversaries (such as the NSA) and adversaries running sophisticated attacks (such as BGP hijacking [29]) may receive the full view of the network traffic, and may also infiltrate the collection of mix-nodes.

Surprisingly, despite its real-world importance, we were the first to consider this question.

Warm-up: An *oblivious permutation algorithm* between a memory-constrained client and an untrusted storage server enables the client to permute a sequence of (encrypted) data blocks stored on the server without the server learning anything (in the statistical sense) about the permutation.

[1] Tor stands for "the onion router", and even though the underlying mechanics are somewhat different from what we described above (instead of using public-key encryption, participants carry out key exchange so that the rest of the communication can be more efficient), the underlying theory is still the same.

[2] Technically, since onion routing uses encryption, the adversary's view cannot be statistically independent of the input, but at best computationally independent. However, as we will see, if we work in an idealized encryption model, such as in Canetti's $\mathcal{F}_{\mathsf{Enc}}$-hybrid model [7], statistical privacy makes sense.

▶ **Theorem 1.** *Any oblivious permutation algorithms can be adapted into a communications protocol for achieving statistical privacy from the network adversary.*

As an example, Ohrimenko et al. [26] presented a family of efficient oblivious permutation algorithms. This can be adapted into a secure and "tunable" OR protocol that can trade off between low server load and latency. Letting λ denote the security parameter, for any $B \in [\frac{\sqrt{N}}{\log^2 \lambda}]$, this protocol can be set to run in $O(\frac{\log N}{\log B})$ rounds with communication complexity overhead $O(\frac{B \log N \log^2 \lambda}{\log B})$ and server load $O(B \log^2 \lambda)$.

However, to be secure from the passive adversary, we need more resources. We prove for the first time that onion routing can provide statistical privacy from the passive adversary, while being efficient.

1. We prove that our solution, Π_p, is statistically private from any passive adversary capable of monitoring any constant $\kappa \in [0, 1)$ of the mix-nodes, while having communication complexity overhead $O(\log^2 \lambda)$, server load $O(\log^2 \lambda)$, and latency $O(\log^2 \lambda)$, where λ denotes the security parameter. (See Section 4.)

However, for most realistic input settings (not constrained to the simple I/O setting), statistical privacy is too ambitious a goal. It is not attainable even with a trusted third party. Following recent literature [3, 31], for our final result, let us *not* restrict users' inputs, and settle for a weaker notion of privacy, namely, differential privacy.

Our definition of differential privacy requires that the difference between the adversary's view when Alice sends a message to Bob and its view when she does not send a message at all or sends it to Carol instead, is small. This is meaningful; showing that the protocol achieves differential privacy gives every user a guarantee that sending her message through does not change the adversary's observations very much.

2. Our solution, Π_a, can defend against the active adversary while having communication complexity overhead $O(\log^6 \lambda)$, server load $O(\log^4 \lambda)$, and latency $O(\log^2 \lambda)$. This is the first provably secure peer-to-peer solution that also provides a level of robustness; unless the adversary forces the honest players to abort the protocol run, most messages that are not dropped by the adversary are delivered to their final destinations. (See Section 5.)

To prepare onions, we use a cryptographic scheme that is strong enough that, effectively, the only thing that the active adversary can do with onions generated by honest parties is to drop them (see the onion cryptosystem by Camenisch and Lysyanskaya [6] for an example of a sufficiently strong cryptosystem). Unfortunately, even with such a scheme, it is still tricky to protect Alice's privacy against an adversary that targets Alice specifically. Suppose that an adversarial Bob is expecting a message of a particular form from an anonymous interlocutor, and wants to figure out if it was Alice or not. If the adversary succeeds in blocking all of Alice's onions and not too many of the onions from other parties, and then Bob never receives the expected message, then the adversary's hunch that it was Alice will be confirmed.

How do we prevent this attack? For this attack to work, the adversary would have to drop a large number of onions – there is enough cover traffic in our protocol that dropping just a few onions does not do much. But once a large enough number of onions is dropped, the honest mix-nodes will detect that an attack is taking place, and will shut down before any onions are delivered to their destinations. Specifically, if enough onions survive half of the rounds, then privacy is guaranteed through having sufficient cover; otherwise, privacy is guaranteed because *no* message reaches its final destination with overwhelming probability. So the adversary does not learn anything about the destination of Alice's onions.

In order to make it possible for the mix-nodes to detect that an attack is taking place, our honest users create "checkpoint" onions. These onions don't carry any messages; instead, they are designed to be "verified" by a particular mix-node in a particular round. These checkpoint onions are expected by the mix-node, so if one of them does not arrive, the mix-node in question realizes that something is wrong. If enough checkpoint onions are missing, the mix-node determines that an attack is underway and shuts down. Two different users, Alice and Allison, use a PRF with a shared key (this shared key need not be pre-computed, but can instead be derived from a discrete-log based public-key infrastructure under the decisional Diffie-Hellman assumption) in order to determine whether Alice should create a checkpoint onion that will mirror Allison's checkpoint onion.

Related work

Encryption schemes that are appropriate for onion routing are known [2,6]. Several papers attempted to define anonymity for communications protocols and to analyze Tor [20,21,28]. Backes et al. [3] were the first to consider a notion inspired by differential privacy [18] but, in analyzing Tor, they assume an adversary with only a partial view of the network. There are also some studies on anonymity protocols, other than onion routing protocols, that were analyzed using information-theoretic measures [1,4,8,16,24]. In contrast, all the protocols presented in this paper are provably secure against powerful adversaries that can observe all network traffic. The system, Vuvuzela [31], assumes that all messages travel through the same set of dedicated servers and is, therefore, impractical compared to Tor. Recently proposed systems, Stadium [30] and Atom [25] are distributed but not robust; they rely on verifiable shuffling to detect and abort. A variant of Atom is robust at a cost in security; it only achieves k-anonymity [25]. In contrast, our solution for the active adversary is distributed while maintaining low latency, and robust while being provably secure.

Achieving anonymous channels using heavier cryptographic machinery has been considered also. One of the earliest examples is Chaum's dining cryptographer's protocol [9]. Rackoff and Simon [27] use secure multiparty computation for providing security from active adversaries. Other cryptographic tools used in constructing anonymity protocols include oblivious RAM (ORAM) and private information retrieval (PIR) [11,12]. Corrigan-Gibbs et al.'s Riposte solution makes use of a global bulletin board and has a latency of a couple of days [12]. The aforementioned Stadium [25] is another solution for a public forum. Blaze et al. [5] provided an anonymity protocol in the wireless (rather than point-to-point) setting.

2 Preliminaries

Notation

By the notation $[n]$, we mean the set $\{1, \ldots, n\}$ of integers. The output a of an algorithm A is denoted by $a \leftarrow A$. For a set S, we write $s \leftarrow S$ to represent that s is a uniformly random sample from the set S and $|S|$, to represent its cardinality. A realization d of a distribution D is denoted $d \sim D$; by $d \sim \mathbf{Binomial}(N, p)$, we mean that d is a realization of a binomial random variable with parameters N and p. By $\log(n)$, we mean the logarithm of n, base 2; and by $\ln(n)$, we mean the natural log of n.

A function $f : \mathbb{N} \to \mathbb{R}$ is negligible in λ, written $f(\lambda) = \mathbf{negl}(\lambda)$, if for every polynomial $p(\cdot)$ and all sufficiently large λ, $f(\lambda) < 1/p(\lambda)$. When λ is the security parameter, we say that an event occurs with overwhelming probability if it is the complement of an event with probability negligible in λ. Two families of distributions $\{D_{0,\lambda}\}_{\lambda\in\mathbb{N}}$ and $\{D_{1,\lambda}\}_{\lambda\in\mathbb{N}}$ are statistically close if the statistical distance between $D_{0,\lambda}$ and $D_{1,\lambda}$ is negligible in λ; we abbreviate this notion by $D_0 \approx_s D_1$ when the security parameter is clear by context. We use the standard notion of a pseudorandom function [22, Ch. 3.6].

Onion routing

Following Camenisch and Lysyanskaya's work on cryptographic onions [6], an *onion routing scheme* is a triple of algorithms: $(\mathsf{Gen}, \mathsf{FormOnion}, \mathsf{ProcOnion})$. The algorithm, Gen, generates a public-key infrastructure for a set of parties. The algorithm, $\mathsf{FormOnion}$, forms onions; and the algorithm, $\mathsf{ProcOnion}$, processes onions.

Given a set $[N]$ of parties, for every $i \in [N]$, let $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{Gen}(1^\lambda)$ be the key pair generated for party $i \in [N]$, where λ denotes the security parameter.

$\mathsf{FormOnion}$ takes as input: a message m, an ordered list $(P_1, \ldots, P_{L+1})$ of parties from $[N]$, and the public-keys $(\mathsf{pk}_{P_1}, \ldots, \mathsf{pk}_{P_{L+1}})$ associated with these parties, and a list $(s_1, \ldots, s_L)$ of (possibly empty) strings that are nonces associated with layers of the onion. The party P_{L+1} is interpreted as the *recipient* of the message, and the list $(P_1, \ldots, P_{L+1})$ is the *routing path* of the message. The output of $\mathsf{FormOnion}$ is a sequence $(O_1, \ldots, O_{L+1})$ of onions. Because it is convenient to think of an onion as a layered encryption object, where processing an onion O_r produces the next onion O_{r+1}, we sometimes refer to the process of revealing the next layer of an onion as "decrypting the onion", or "peeling the onion". For every $r \in [L]$, only party P_r can peel onion O_r to reveal the next layer, $(P_{r+1}, O_{r+1}, s_{r+1}) \leftarrow \mathsf{ProcOnion}(\mathsf{sk}_{P_r}, O_r, P_r)$, of the onion containing the "peeled" onion O_{r+1}, the "next destination" P_{r+1}, and the nonce s_{r+1}. Only the recipient P_{L+1} can peel the innermost onion O_{L+1} to reveal the message, $m \leftarrow \mathsf{ProcOnion}(\mathsf{sk}_{P_{L+1}}, O_{L+1}, P_{L+1})$.

Let O_0 be an onion formed from running $\mathsf{FormOnion}(m_0, P^0, \mathsf{pk}^0, s^0)$, and let O_1 be another onion formed from running $\mathsf{FormOnion}(m_1, P^1, \mathsf{pk}^1, s^1)$. Importantly, a party that can't peel either onion can't tell which input produced which onion. See Camenisch and Lysyanskaya's paper [6] for formal definitions.

In our protocols, a sender of a message m to a recipient j "forms an onion" by generating nonces and running the $\mathsf{FormOnion}$ algorithm on the message m, a routing path $(P_1, \ldots, P_L, j)$, the public keys $(\mathsf{pk}_{P_1}, \ldots, \mathsf{pk}_{P_L}, \mathsf{pk}_j)$ associated with the parties on the routing path, and the generated nonces; the "formed onion" is the first onion O_1 from the list of outputted onions. The sender sends O_1 to the first party P_1 on the routing path, who processes it and sends the peeled onion O_2 to the next destination P_2, and so on, until the last onion O_{L+1} is received by the recipient j, who processes it to obtain the message m.

3 Definitions

We model the network as a graph with N nodes, and we assume that these nodes are synchronized. This way, any onion can be sent from any sender to any receiver, and also its transmission occurs within a single round.

Every participant is a user client, and some user clients also serve as mix-nodes. In all the definitions, the N users participating in an communications protocol Π are labeled $1, \ldots, N$; and the number N of users is assumed to be polynomially-bounded in the security parameter λ. Every input to a protocol is an N-dimensional vector. When a protocol runs on input $\sigma = (\sigma_1, \ldots, \sigma_N)$, it means that the protocol is instantiated with each user i receiving σ_i as input. $\mathcal{M}$ denotes the (bounded) message space. A message pair (m, j) is properly formed if $m \in \mathcal{M}$ and $j \in [N]$. The input σ_i to each user $i \in [N]$ is a collection of properly formed message pairs, where $(m, j) \in \sigma_i$ means that user i intends on sending message m to user j. Let $M(\sigma)$ denote the "messages in σ". It is the multiset of all message pairs in σ, that is $M(\sigma_1, \ldots, \sigma_N) = \bigcup_{i=1}^{N} \{(m, j) \in \sigma_i\}$.

For analyzing our solutions, it is helpful to first assume an idealized version of an encryption scheme, in which the ciphertexts are information-theoretically unrelated to the plaintexts that they encrypt and reveal nothing but the length of the plaintext. Obviously, such encryption schemes do not exist computationally, but only in a hybrid model with an oracle that realizes an ideal encryption functionality, such as that of Canetti [7]. When used in forming onions, such an encryption scheme gives rise to onions that are information-theoretically independent of their contents, destinations, and identities of the mix-nodes. Our real-life proposal, of course, will use standard computationally secure encryption [17]. We discuss the implications of this in the full version of this paper.

Views and outputs

We consider the following standard adversary models, in increasing order of capabilities:

1. **Network adversary.** A network adversary can observe the bits flowing on every link of the network. (Note that if the peer-to-peer links are encrypted in an idealized sense, then the only information that the adversary can use is the volume flow.)
2. **Passive adversary.** In addition to the capabilities of a network adversary, a passive adversary can monitor the internal states and operations of a constant fraction of the parties. The adversary's choices for which parties to monitor are made non-adaptively over the course of the execution run.
3. **Active adversary.** In addition to the capabilities of a network adversary, an active adversary can corrupt a constant fraction of the parties. The adversary's choices for which parties to corrupt are made non-adaptively over the course of the execution run. The adversary can change the behavior of corrupted parties to deviate arbitrarily from the protocol.

Let Π be a protocol, and let σ be a vector of inputs to Π. Given an adversary $\mathcal{A}$, the view $V^{\Pi,\mathcal{A}}(\sigma)$ of $\mathcal{A}$ is its observables from participating in Π on input σ plus any randomness used to make its decisions. With idealized secure peer-to-peer links, the observables for a network adversary are the traffic volumes on all links; whereas for the passive and active adversaries, the observables additionally include the internal states and computations of all monitored / corrupted parties at all times.

Given an adversary $\mathcal{A}$, the output $O^{\Pi,\mathcal{A}}(\sigma) = (O_1^{\Pi,\mathcal{A}}(\sigma), \ldots, O_N^{\Pi,\mathcal{A}}(\sigma))$ of Π on input σ is a vector of outputs for the N parties.

3.1 Privacy definitions

How do we define security for an anonymous channel? The adversary's view also includes the internal states of corrupted parties. In such case, we may wish to protect the identities of honest senders from the recipients that are in cahoots with the adversary. However, even an ideal anonymous channel cannot prevent the contents of messages (including the volumes of messages) from providing a clue on who sent the messages; thus any "message content" leakage should be outside the purview of an anonymous channel. To that end, we say that an communications protocol is secure if it is difficult for the adversary to learn who is communicating with whom, beyond what leaks from captured messages.

Below, we provide two flavors of this security notion; we will prove that our constructions achieve either statistical privacy or (ϵ, δ)-differential privacy [19, Defn. 2.4] in the idealized encryption setting.

▶ **Definition 2** (Statistical privacy). Let Σ^* be the input set consisting of every input of the form $\sigma = (\{(m_1, \pi(1))\}, \dots, \{(m_N, \pi(N))\})$, where $\pi : [N] \to [N]$ is any permutation function over the set $[N]$, and $m_i \in \mathcal{M}$ for every $i \in [N]$. A communications protocol Π is *statistically private* from every adversary from the class $\mathbb{A}$ if for all $\mathcal{A} \in \mathbb{A}$ and for all $\sigma_0, \sigma_1 \in \Sigma^*$ that differ only on the honest parties' inputs and outputs, the adversary's views $V^{\Pi,\mathcal{A}}(\sigma_0)$ and $V^{\Pi,\mathcal{A}}(\sigma_1)$ are statistically indistinguishable, i.e., $\Delta(V^{\Pi,\mathcal{A}}(\sigma_0), V^{\Pi,\mathcal{A}}(\sigma_1)) = \mathbf{negl}(\lambda)$, where $\lambda \in \mathbb{N}$ denotes the security parameter, and $\Delta(\cdot,\cdot)$ denotes statistical distance (i.e., total variation distance). Π is perfectly secure if the statistical distance is zero instead.

▶ **Definition 3** (Distance between inputs). The *distance* $d(\sigma_0, \sigma_1)$ between two inputs $\sigma_0 = (\sigma_{0,1}, \dots, \sigma_{0,N})$ and $\sigma_1 = (\sigma_{1,1}, \dots, \sigma_{1,N})$ is given by $d(\sigma_0, \sigma_1) \overset{\text{def}}{=} \sum_{i=1}^{N} |\sigma_{0,i} \nabla \sigma_{1,i}|$, where $(\cdot\nabla\cdot)$ denotes the symmetric difference.

▶ **Definition 4** (Neighboring inputs). Two inputs σ_0 and σ_1 are *neighboring* if $d(\sigma_0, \sigma_1) \leq 1$.

▶ **Definition 5** ((ϵ, δ)-DP [19, Defn. 2.4]). Let Σ be the set of all valid inputs. A communications protocol is (ϵ, δ)*-DP* from every adversary in the class $\mathbb{A}$ if for all $\mathcal{A} \in \mathbb{A}$, for every neighboring inputs $\sigma_0, \sigma_1 \in \Sigma$ that differ only on an honest party's input and an honest party's output, and any set $\mathcal{V}$ of views, $\Pr[V^{\Pi,\mathcal{A}}(\sigma_0) \in \mathcal{V}] \leq e^{\epsilon} \cdot \Pr[V^{\Pi,\mathcal{A}}(\sigma_1) \in \mathcal{V}] + \delta$.

While differential privacy is defined with respect to neighboring inputs, it also provides (albeit weaker) guarantees for non-neighboring inputs; it is known that the security parameters degrade proportionally in the distance between the inputs [19].

3.2 Other performance metrics

Since message delivery cannot be guaranteed in the presence of an active adversary, we define correctness with respect to *passive* adversaries.

▶ **Definition 6** (Correctness). A communications protocol Π is *correct* on an input $\sigma \in \Sigma$ if for any passive adversary $\mathcal{A}$, and for every recipient $j \in [N]$, the output $O_j^{\Pi,\mathcal{A}}(\sigma)$ corresponds to the multiset of all messages for recipient j in the input vector σ. That is, $O_j^{\Pi,\mathcal{A}}(\sigma) = \{m \,|(m,j) \in M(\sigma)\}$, where $M(\sigma)$ denotes the multiset of all messages in σ.

Efficiency of OR protocols

The communication complexity blow-up of an onion routing (OR) protocol measures how many more onion transmissions are required by the protocol, compared with transmitting the messages in onions directly from the senders to the recipients (without passing through intermediaries). We assume that every message $m \in \mathcal{M}$ in the message space $\mathcal{M}$ "fits" into a single onion. The communication complexity is measured in unit onions, which is appropriate when the parties pass primarily onions to each other.

▶ **Definition 7** (Communication complexity blow-up). The *communication complexity blow-up* of an OR protocol Π is defined with respect to an input vector σ and an adversary $\mathcal{A}$. Denoted $\gamma^{\Pi,\mathcal{A}}(\sigma)$, it is the expected ratio between the total number $\Gamma^{\Pi,\mathcal{A}}(\sigma)$ of onions transmitted in protocol Π and the total number $|M(\sigma)|$ of messages in the input vector. That is, $\gamma^{\Pi,\mathcal{A}}(\sigma) \overset{\text{def}}{=} \mathbb{E}\left[\frac{\Gamma^{\Pi,\mathcal{A}}(\sigma)}{|M(\sigma)|}\right]$.

▶ **Definition 8** (Server load). The *server load* of an OR protocol Π is defined with respect to an input vector σ and an adversary $\mathcal{A}$. It is the expected number of onions processed by a single party in a round.

▶ **Definition 9** (Latency). The *latency* of an OR protocol Π is defined with respect to an input vector σ and an adversary $\mathcal{A}$. It is the expected number of rounds in a protocol execution.

In addition to having low (i.e., polylog in the security parameter) communication complexity blow-ups, we will show that our OR protocols have low (i.e., polylog in the security parameter) server load and low (i.e., polylog in the security parameter) latency.

4 The passive adversary

Communication patterns can trivially be hidden by sending every message to every participant in the network, but this solution is not scalable as it requires a communication complexity blow-up that is linear in the number of participants. Here, we prove that an OR protocol can provide anonymity from the passive adversary while being practical with low communication complexity and low server load.

To do this, every user must send and receive the same number of messages as any other user; otherwise, the sender-receiver relation can leak from the differing volumes of messages sent and received by the users. In other words, every user essentially commits to sending a message, be it the empty message $\perp$ to itself. Let Σ^* be the set of all input vectors of the form $\sigma = (\{(m_1, \pi(1))\}, \dots, \{(m_N, \pi(N))\})$, where $\pi : [N] \to [N]$ is any permutation function over the set $[N]$, and $m_1, \dots, m_N$ are any messages from the message space $\mathcal{M}$; our solution, Π_p, is presented in the setting where the input vector is constrained to Σ^*.

Let $[N]$ be the set of users, and $\mathcal{S} = \{S_1, \dots, S_n\} \subset [N]$ the set of servers. Π_p uses a secure onion routing scheme, denoted by $\mathcal{OR} = (\mathsf{Gen}, \mathsf{FormOnion}, \mathsf{ProcOnion})$, as a primitive building block. For every $i \in [N]$, let $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{Gen}(1^\lambda)$ be the key pair generated for party i, where λ denotes the security parameter.

During a setup phase, each user $i \in [N]$ creates an onion. On input $\sigma_i = \{(m, j)\}$, user i first picks a sequence $T_1, \dots, T_L$ servers, where each server is chosen independently and uniformly at random, and then forms an onion from the message m, the routing path $(T_1, \dots, T_L, j)$, the public keys $(\mathsf{pk}_{T_1}, \dots, \mathsf{pk}_{T_L}, \mathsf{pk}_j)$ associated with the parties on the routing path, and a list of empty nonces. At the first round of the protocol run, user i sends the formed onion to the first hop T_1 on the routing path.

After every round $i \in [L]$ (but before round $i+1$) of the protocol run, each server processes the onions it received at round i. At round $i+1$, the resulting peeled onions are sent to their respective next destinations in random order. At round $L+1$, every user receives an onion and processes it to reveal a message.

Correctness and efficiency

Clearly, Π_p is correct. In Π_p, N messages are transmitted in each of the $L+1$ rounds of the protocol run. Thus, the communication complexity blow-up and the latency are both $L+1$. The server load is $\frac{N}{n}$.

Privacy

To prove that Π_p is statistically private from the passive adversary, we first prove that it is secure from the network adversary.

▶ **Theorem 10.** *Π_p is statistically private from the network adversary when $\frac{N}{n} = \Omega(\log^2 \lambda)$, and $L = \Omega(\log^2 \lambda)$, where $\lambda \in \mathbb{N}$ denotes the security parameter.*

Proof. Let $\frac{N}{n} = \alpha \log^2 \lambda$; so that after each round, every location receives $\alpha \log^2 \lambda$ onions in expectation. We recast our problem as a balls-in-bins problem, where the balls are the onions, and the bins, the locations. At every round of the protocol, all $\alpha n \log^2 \lambda$ balls (i.e., onions) are thrown uniformly at random into n bins (i.e., each onion is routed to one of n locations, chosen independently and uniformly at random).

Fix any target sender U, and let $X^r = (X_1^r, \dots, X_n^r)$ be a vector of non-negative numbers summing to one, representing $\mathcal{A}$'s best estimate for the location of U's ball after r rounds (and before round $r+1$); for every $i \in [n]$, X_i^r is the likelihood that bin i contains U's ball after r rounds. Let $(X^r_{f_r(1)}, \dots, X^r_{f_r(n)})$ be the result of sorting $(X_1^r, \dots, X_n^r)$ in decreasing order, where $f_r : [n] \to [n]$ is a permutation function over the set $[n]$. For every $i \in [n]$, let $b_i^r = f_r(i)$ be the index of the bin with the i-th largest likelihood at round r.

W.l.o.g. we assume that n is divisible by three. We partition the bins into three groups G_1^r, G_2^r, and G_3^r; such that G_1^r contains all the *balls* in the top one-third most likely bins $b_1^r, \dots, b^r_{\frac{n}{3}}$; G_3^r contains all the balls in the bottom one-third most likely bins $b^r_{\frac{2n}{3}+1}, \dots, b_n^r$; and G_2^r contains all the balls in the remaining bins $b^r_{\frac{n}{3}+1}, \dots, b^r_{\frac{2n}{3}}$.

For each $j \in [3]$, let $O_j^r \in G_j^r$ be a ball with the maximum likelihood of being U's onion among the balls in group G_j^r. For any $d \in (0,1)$, let $d' = 1 - \frac{1}{1+d}$. Let c_j^r be the bin containing O_j^r. The bin c_j^r contains at least $(1-d')\alpha \log^2 \lambda$ balls (Chernoff bounds for Poisson trials). It follows that,

$$\Pr[O_j^r \text{ is } U\text{'s onion}] \le (1+d)\frac{X^r_{c_j^r}}{\alpha \log^2 \lambda} \le (1+d)\frac{X^r_{\frac{(j-1)n}{3}+1}}{\alpha \log^2 \lambda} \tag{1}$$

with overwhelming probability, where $X^r_{\frac{(j-1)n}{3}+1}$ is the likelihood of the most likely bin in group G_j.

The number of balls contained in each group G_j^r is arbitrarily close to the expected number $\frac{\alpha}{3} n \log^2 \lambda$ of balls in a group (Chernoff bounds). Thus, the most probable bin b_1^{r+1} after the next round receives at most $\frac{(1+d)\alpha}{3} \log^2 \lambda$ balls from each of the three groups: G_1^r, G_2^r, and G_3^r. From (1), this implies that, with overwhelming probability, $X_1^{r+1} \le \frac{(1+d)^2}{3} \sum_{j=1}^3 X^r_{\frac{(j-1)n}{3}+1}$. Using a symmetric argument, we can conclude that, with overwhelming probability, $X_n^{r+1} \ge \frac{(1-d)^2}{3} \sum_{j=1}^3 X^r_{\frac{jn}{3}}$, where $X^r_{\frac{jn}{3}}$ is the likelihood of the least likely bin in group G_j.

For all $r \in [L]$, define $g^r = X_1^r - X_n^r$ as the difference in likelihoods between the most and least likely bins at round r.

$$g^{r+1} \le \frac{(1+d)^2 \sum_{j=1}^3 X^r_{\frac{(j-1)n}{3}+1}}{3} - \frac{(1-d)^2 \sum_{j=1}^3 X^r_{\frac{jn}{3}}}{3} \le \frac{1}{2}\left(X_1^r - X_n^r\right) = \frac{g^r}{2},$$

where the latter inequality follows from telescopic cancelling, since $X^r_{\frac{n}{3}+1} \le X^r_{\frac{n}{3}}$, and $X^r_{\frac{2n}{3}+1} \le X^r_{\frac{2n}{3}}$.

The difference g^r is at least halved at every round. By round $\log^2 \lambda$, the difference is negligible in λ. Thus, after traveling L random hops, each onion becomes unlinked from its sender. Since everyone sends the same number of onions, and everyone receives the same number of onions; it follows that the adversary's views from any two inputs are statistically indistinguishable.

In the proof above, the bins were partitioned into three groups at every round. By partitioning the bins into an appropriately large constant number of groups, we can show that Π_p achieves statistical privacy after $L = \Omega(\log^2 \lambda)$ rounds. ◀

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

▶ **Theorem 11.** *Π_p is statistically private from the passive adversary capable of monitoring any constant fraction $\kappa \in [0, 1)$ of the servers when $\frac{N}{n} = \Omega(\log^2 \lambda)$, and $L = \Omega(\log^2 \lambda)$, where $\lambda \in \mathbb{N}$ denotes the security parameter.*

Proof. We prove this by cases.

In the first case, σ_1 is the same as σ_0 except that the inputs of two users are swapped, i.e., $d(\sigma_0, \sigma_1) = 2$. Using Chernoff bounds for Poisson trials, there are at least some polylog number of rounds where the swapped onions are both routed to an honest bin (not necessarily the same bin). From Theorem 10, after the polylog number of steps, the locations of these two target onions are statistically indistinguishable from each other.

In the second case, $d(\sigma_0, \sigma_1) > 2$. However, the distance between σ_0 and σ_1 is always polynomially bounded. By a simple hybrid argument, it follows that $V^{\Pi,\mathcal{A}}(\sigma_0) \approx_s V^{\Pi,\mathcal{A}}(\sigma_1)$ from case 1. ◀

Remark: Protocol Π_p is not secure from the active adversary. This is because, with non-negligible probability, any honest user will choose a corrupted party as its first hop on its onion's routing path, in which case the adversary can drop the target user's onion at the first hop and observe who does not receive an onion at the last round.

5 The active adversary

We now present an OR protocol, Π_a, that is secure from the active adversary. The setting for Π_a is different from that of our previous solution in a couple of important ways. Whereas Π_p is *statistically* private from the passive adversary, Π_a is only *differentially* private from the active adversary. The upside is that we are no longer constrained to operate in the simple I/O setting; the input can be any valid input.

We let $[N]$ be the set of N parties participating in a protocol. Every party is both a user and a server. As before, $\mathcal{OR} = (\mathsf{Gen}, \mathsf{FormOnion}, \mathsf{ProcOnion})$ is a secure onion routing scheme; and for every $i \in [N]$, $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{Gen}(1^\lambda)$ denotes the key pair generated for party i, where λ is the security parameter. Further, we assume that every pair $(i, k) \in [N]^2$ of parties shares a common secret key[3], denoted by $\mathsf{sk}_{i,k}$. F is a pseudorandom function (PRF).

We describe the protocol by the setup and routing algorithms for party $i \in [N]$; each honest party runs the same algorithms.

Setup

Let $L = \beta \log^2 \lambda$ for some constant $\beta > 0$. During the setup phase, party i prepares a set of onions from its input. For every message pair $u = \{m, j\}$ in party i's input, party i picks a sequence $T_1^u, \ldots, T_L^u$ of parties, where each party T_ℓ^u is chosen independently and uniformly at random, and forms an onion from the message m, the routing path $(T_1^u, \ldots, T_L^u, j)$, the public keys $(\mathsf{pk}_{T_1^u}, \ldots, \mathsf{pk}_{T_L^u}, \mathsf{pk}_j)$, and a list of empty nonces.

Additionally, party i forms some dummy onions, where a dummy onion is an onion formed using the empty message $\perp$.

```
1: for every index $(r, k) \in [L] \times [N]$:
2:     compute $b \leftarrow F(\mathsf{sk}_{i,k}, \mathsf{session} + r, 0)$, where $\mathsf{session} \in \mathbb{N}$ denotes the protocol instance.
3:     if $b \equiv 1$ – set to occur with frequency $\frac{\alpha \log^2 \lambda}{N}$ for some constant $\alpha > 0$ – do:
```

[3] In practice, the shared keys do not need to be set up in advance; they can be generated as needed from an existing PKI, e.g., using Diffie-Hellman.

```
4:    choose a list $T^{r,k} = (T_1^{r,k}, \ldots, T_{r-1}^{r,k}, T_{r+1}^{r,k}, \ldots, T_{L+1}^{r,k})$ parties, where each party is
      chosen independently and uniformly at random;
5:    create a list $s^{r,k} = (s_1^{r,k}, \ldots, s_2^{r,k})$ of nonces, where
      $s_r^{r,k} = (\mathsf{checkpt}, F(\mathsf{sk}_{i,k}, \mathsf{session} + r, 1))$, and all other elements of $s^{r,k}$ are $\perp$; and
6:    form a dummy onion using the message $\perp$, the routing path $T^{r,k}$, the public keys
      associated with $T^{r,k}$, and the list $s^{r,k}$ of nonces. end if, end for
```

The additional information $s_r^{r,k}$ is embedded in only the r-th layer; no additional information is embedded in any other layer. At the first round of the protocol run (after setup), all formed onions are sent to their respective first hops.

Routing

If party i forms a dummy onion with nonce $s_r^{r,k}$ embedded in the r-th layer, then it expects to receive a symmetric dummy onion at the r-th round formed by another party k that, when processed, reveals the same nonce $s_r^{r,k}$. If many checkpoint nonces are missing, then party i knows to abort the protocol run.

After every round $r \in [L]$ (but before round $r+1$), party i peels the onions it received at round r and counts the number of missing checkpoint nonces. If the count exceeds a threshold value t, the party aborts the protocol run; otherwise, at round $r+1$, the peeled onions are sent to their next destinations in random order. After the final round, party i outputs the set of messages revealed from processing its the onions it receives at round $L+1$.

Correctness and efficiency

Recalling that correctness is defined with respect to the passive adversary, Π_a is clearly correct. Moreover, unless an honest party aborts the protocol run, all messages that are not dropped by the adversary are delivered to their final destinations. In Π_a, the communication complexity blow-up is $O(\log^6 \lambda)$, since the latency is $L + 1 = O(\log^2 \lambda)$ rounds, and the server load is $O(\log^4 \lambda)$.

Privacy

To prove that Π_a is secure, we require that the thresholding mechanism does its job:

▶ **Lemma 12.** *In Π_a, if F is a random function, $t = c(1-d)(1-\kappa)^2 \alpha \log^2 \lambda$ for some $c, d \in (0,1)$, and an honest party does not abort within the first r rounds of the protocol run, then with overwhelming probability, at least $(1-c)$ of the dummy onions created between honest parties survive at least $(r-1)$ rounds, even in the presence of an active adversary non-adaptively corrupting a constant fraction $\kappa \in [0,1)$ of the parties.*

The proof relies on a known concentration bound for the hypergeometric distribution [23] and can be found in the full version of this paper.

▶ **Theorem 13.** *If, in Π_a, F is a random function, $N \geq \frac{3}{1-\kappa}$, and $t = c(1-d)(1-\kappa)^2 \alpha \log^2 \lambda$ for some $c, d \in (0,1)$, then, for $\alpha\beta \geq -\frac{36(1+\epsilon/2)^2 \ln(\delta/4)}{(1-c)(1-\kappa)^2 \epsilon^2}$, Π_a is (ϵ, δ)-DP from the active adversary non-adaptively corrupting a constant fraction $\kappa \in [0,1)$ of the parties.*

Proof. The proof is by cases.

Case 1: All honest parties abort within the first half of the protocol run. With overwhelming probability, no onion created by an honest party will be delivered to its final destination (Chernoff bounds for Poisson trials), and so the adversary doesn't learn anything.

Case 2: Some honest party doesn't abort within the first half of the protocol run. Let $\mathcal{A}$ be any adversary that non-adaptively corrupts a constant $\kappa \in [0,1)$ of the parties. Suppose that for every onion that survive the first half of the protocol run, a dark angel provides the adversary $\mathcal{A}$ with the second half of the onion's routing path. Further suppose that no other onions are dropped in the second half of the protocol run. (If more onions are dropped, then Π_a is secure from the post-processing theorem for differential privacy [19, Proposition 2.1].)

For any two neighboring inputs σ_0 and σ_1, the only difference in the adversary's views, $V^{\Pi_a,\mathcal{A}}(\sigma_0)$ and $V^{\Pi_a,\mathcal{A}}(\sigma_1)$, is the routing of a single onion O. If there is an honest party who does not abort within the first half of the protocol run, then from Lemma 12, some constant fraction of the dummy onions created by the honest parties survive the first half of the protocol run with overwhelming probability. So, from Theorem 11, the onions are no longer linked to their senders by the end of the first half of the protocol run. Thus, the only information that $\mathcal{A}$ could find useful is the volume of onions sent out by the sender P_s of the extra onion O and the volume of onions received by the receiver P_r of O.

Let X denote the number of dummy onions created by P_s. For every $(k,r) \in [L] \times [N]$, an honest sender P_s creates a dummy onion with probability $\frac{\alpha \log^2 \lambda}{N}$; so $X \sim \textbf{Binomial}(H,p)$, where $H = LN$, and $p = \frac{\alpha \log^2 \lambda}{N}$.

Let $Y \sim \textbf{Binomial}(G,q)$ be another binomial random variable with parameters $G = \frac{L(1-\kappa)^2 N^2}{3}$ and $q = \frac{(1-c)\alpha \log^2 \lambda}{N^2}$. For $N \geq \frac{3}{1-\kappa}$ and sufficiently small $d > 0$, $G \leq (1-d)L\binom{(1-\kappa)N-1}{2}$; thus, with overwhelming probability, Y is less than the number of dummy onions created between honest non-P_s parties and received by P_r in the final round (Chernoff bounds).

Let $\mathcal{O} \overset{\text{def}}{=} \mathbb{N} \times \mathbb{N}$ be the sample space for the multivariate random variable (X,Y).

Let $\mathcal{O}_1$ be the event that $|X - \mathbb{E}[X]| \leq d'\mathbb{E}[X]$, and $|Y - \mathbb{E}[Y]| \leq d'\mathbb{E}[Y]$, where $d' = \frac{\epsilon/2}{1+\epsilon/2}$, $\mathbb{E}[X] = Hp$ is the expected value of X, and $\mathbb{E}[Y] = Gq$ is the expected value of Y; and let $\bar{\mathcal{O}}_1$ be the complement of $\mathcal{O}_1$.

For every $(x,y) \in \mathcal{O}_1$, we can show that

$$\max\left(\frac{\Pr[(X,Y)=(x,y)]}{\Pr[(X,Y)=(x+1,y+1)]}, \frac{\Pr[(X,Y)=(x+1,y+1)]}{\Pr[(X,Y)=(x,y)]}\right) \leq e^{\epsilon}. \tag{2}$$

We can also show that the probability of the tail event $\bar{\mathcal{O}}_1$ occurring is negligible in λ and at most δ when $\alpha\beta \geq -\frac{36(1+\epsilon/2)^2 \ln(\delta/4)}{(1-c)(1-\kappa)^2\epsilon^2}$. (See the full version of this paper.)

Any event $\mathcal{E}$ can be decomposed into two subsets $\mathcal{E}_1$ and $\mathcal{E}_2$, such that (1) $\mathcal{E} = \mathcal{E}_1 \cup \mathcal{E}_2$, (2) $\mathcal{E}_1 \subseteq \mathcal{O}_1$, and (3) $\mathcal{E}_2 \subseteq \bar{\mathcal{O}}_1$. It follows that, for every event $\mathcal{E}$,

$$\Pr[(X,Y) \in \mathcal{E}] \leq e^{\epsilon} \cdot \Pr[(X+1,Y+1) \in \mathcal{E}] + \delta, \text{ and} \tag{3}$$

$$\Pr[(X+1,Y+1) \in \mathcal{E}] \leq e^{\epsilon} \cdot \Pr[(X,Y) \in \mathcal{E}] + \delta. \tag{4}$$

The views $V^{\Pi_a,\mathcal{A}}(\sigma_0)$ and $V^{\Pi_a,\mathcal{A}}(\sigma_1)$ are the same except that O exists in one of the views but not in the other. Thus, (3) and (4) suffice to show that for any set $\mathcal{V}$ of views and for any $b \in \{0,1\}$, $\Pr[V_b^{\Pi_a,\mathcal{A}} \in \mathcal{V}] \leq e^{\epsilon} \cdot \Pr[V_{\bar{b}}^{\Pi_a,\mathcal{A}} \in \mathcal{V}] + \delta$, where $\bar{b} = b + 1 \mod 2$. ◀

Remark: Our protocols are for a *single-pass* setting, where the users send out messages once. It is clear how our statistical privacy results would compose for the multi-pass case. To prove that Π_a also provides differential privacy in the multi-pass scenario – albeit for degraded security parameters – we can use the k-fold composition theorem [19]; the noise falls at a rate of the square-root of the number of runs.

References

1 Mário S. Alvim, Miguel E. Andrés, Konstantinos Chatzikokolakis, Pierpaolo Degano, and Catuscia Palamidessi. Differential privacy: on the trade-off between utility and information leakage. In *FAST 2011*, pages 39–54. Springer, 2011.

2 Michael Backes, Ian Goldberg, Aniket Kate, and Esfandiar Mohammadi. Provably secure and practical onion routing. In *Computer Security Foundations Symposium (CSF), 2012 IEEE 25th*, pages 369–385. IEEE, 2012.

3 Michael Backes, Aniket Kate, Praveen Manoharan, Sebastian Meiser, and Esfandiar Mohammadi. AnoA: A framework for analyzing anonymous communication protocols. Cryptology ePrint Archive, Report 2014/087, 2014. URL: `http://eprint.iacr.org/2014/087`.

4 Ron Berman, Amos Fiat, and Amnon Ta-Shma. Provable unlinkability against traffic analysis. In Ari Juels, editor, *FC 2004*, volume 3110 of *LNCS*, pages 266–280, Key West, USA, feb 9–12, 2004. Springer, Heidelberg, Germany.

5 Matt Blaze, John Ioannidis, Angelos D Keromytis, Tal Malkin, and Aviel D Rubin. Anonymity in wireless broadcast networks. *IJ Network Security*, 8(1):37–51, 2009.

6 Jan Camenisch and Anna Lysyanskaya. A formal treatment of onion routing. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 169–187, Santa Barbara, CA, USA, aug 14–18, 2005. Springer, Heidelberg, Germany.

7 Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145, Las Vegas, NV, USA, oct 14–17, 2001. IEEE Computer Society Press.

8 Konstantinos Chatzikokolakis, Catuscia Palamidessi, and Prakash Panangaden. Anonymity protocols as noisy channels. *Information and Computation*, 206(2–4):378–401, 2008.

9 David Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of Cryptology*, 1(1):65–75, 1988.

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

11 David A. Cooper and Kenneth P. Birman. Preserving privacy in a network of mobile computers. In *1995 IEEE Symposium on Security and Privacy*, pages 26–38. IEEE Computer Society Press, 1995.

12 Henry Corrigan-Gibbs, Dan Boneh, and David Mazières. Riposte: An anonymous messaging system handling millions of users. In *2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, May 17-21, 2015*, pages 321–338, San Jose, CA, USA, may 17–21, 2015. IEEE Computer Society Press. `doi:10.1109/SP.2015.27`.

13 George Danezis, Roger Dingledine, and Nick Mathewson. Mixminion: Design of a type III anonymous remailer protocol. In *2003 IEEE Symposium on Security and Privacy*, pages 2–15, Berkeley, CA, USA, may 11–14, 2003. IEEE Computer Society Press.

14 Roger Dingledine and Nick Mathewson. Tor: An anonymous internet communication system. In *Workshop on Vanishing Anonymity, Proceedings from the Conference on Computers, Freedom, and Privacy*, 2005.

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

16 Yevgeniy Dodis, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 523–540, Interlaken, Switzerland, may 2–6, 2004. Springer, Heidelberg, Germany.

17 Danny Dolev, Cynthia Dwork, and Moni Naor. Nonmalleable cryptography. *SIAM Journal on Computing*, 30(2):391–437, 2000.

18 Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In Shai Halevi and Tal Rabin, editors, *TCC 2006*,

volume 3876 of *LNCS*, pages 265–284, New York, NY, USA, mar 4–7, 2006. Springer, Heidelberg, Germany.

19 Dwork, Cynthia and Roth, Aaron. The Algorithmic Foundations of Differential Privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014.

20 Joan Feigenbaum, Aaron Johnson, and Paul Syverson. Probabilistic analysis of onion routing in a black-box model. *ACM Transactions on Information and System Security (TISSEC)*, 15(3):1–28, Nov 2012.

21 Joan Feigenbaum, Aaron Johnson, and Paul F. Syverson. A model of onion routing with provable anonymity. In Sven Dietrich and Rachna Dhamija, editors, *FC 2007*, volume 4886 of *LNCS*, pages 57–71, Scarborough, Trinidad and Tobago, feb 12–16, 2007. Springer, Heidelberg, Germany.

22 Oded Goldreich. *Foundations of Cryptography: Basic Tools*, volume 1. Cambridge University Press, Cambridge, UK, 2001.

23 Don Hush and Clint Scovel. Concentration of the hypergeometric distribution. *Statistics & probability letters*, 75(2):127–132, 2005.

24 Boris Köpf and David A. Basin. An information-theoretic model for adaptive side-channel attacks. In Peng Ning, Sabrina De Capitani di Vimercati, and Paul F. Syverson, editors, *ACM CCS 07*, pages 286–296, Alexandria, Virginia, USA, oct 28–31, 2007. ACM Press.

25 Albert Kwon, Henry Corrigan-Gibbs, Srinivas Devadas, and Bryan Ford. Atom: Horizontally scaling strong anonymity. In *Proceedings of the 26th Symposium on Operating Systems Principles*, SOSP '17, pages 406–422, New York, NY, USA, 2017. ACM. doi:10.1145/3132747.3132755.

26 Olga Ohrimenko, Michael T. Goodrich, Roberto Tamassia, and Eli Upfal. The melbourne shuffle: Improving oblivious storage in the cloud. In Javier Esparza, Pierre Fraigniaud, Thore Husfeldt, and Elias Koutsoupias, editors, *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part II*, volume 8573 of *LNCS*, pages 556–567, Copenhagen, Denmark, jul 8–11, 2014. Springer, Heidelberg, Germany. doi:10.1007/978-3-662-43951-7_47.

27 Charles Rackoff and Daniel R. Simon. Cryptographic defense against traffic analysis. In *25th ACM STOC*, pages 672–681, San Diego, CA, USA, may 16–18, 1993. ACM Press.

28 Vitaly Shmatikov and Ming-Hsiu Wang. Measuring relationship anonymity in mix networks. In *Proceedings of the 5th ACM workshop on Privacy in electronic society*, pages 59–62. ACM, 2006.

29 Yixin Sun, Anne Edmundson, Nick Feamster, Mung Chiang, and Prateek Mittal. Counter-RAPTOR: Safeguarding tor against active routing attacks. In *2017 IEEE Symposium on Security and Privacy*, pages 977–992, San Jose, CA, USA, may 22–26, 2017. IEEE Computer Society Press.

30 Nirvan Tyagi, Yossi Gilad, Derek Leung, Matei Zaharia, and Nickolai Zeldovich. Stadium: A distributed metadata-private messaging system. In *Proceedings of the 26th Symposium on Operating Systems Principles*, SOSP '17, pages 423–440, New York, NY, USA, 2017. ACM. doi:10.1145/3132747.3132783.

31 Jelle van den Hooff, David Lazar, Matei Zaharia, and Nickolai Zeldovich. Vuvuzela: scalable private messaging resistant to traffic analysis. In *SOSP 2015*, pages 137–152. ACM Press, 2015.