Traffic Analysis by Adversaries with Partial Visibility

Iness Ben Guirat[1][0000−0002−8766−594X], Claudia Diaz[1][0000−0003−2336−7123], Karim Eldefrawy[2][0000−0002−4008−0047], and Hadas Zeilberger[3][0009−0005−5633−0126]

[1] COSIC, KU Leuven, Belgium
{ibenguir,cdiaz}@esat.kuleuven.be
[2] SRI, USA karim.eldafrawi@sri
[3] YALE, USA hadas.zeilberger@yale.com

Abstract. Mixnets are a fundamental privacy enhancing technology in the context of anonymous communication that have been extensively studied in terms of a Global passive Adversary (GPA). However a more realistic adversary that captures only a portion of the network have not yet been studied. We call these adversaries "Adversaries with Partial Visibility (APV)" In this paper we provide a framework that models the mixnet-based system and captures different types of Adversaries with Partial Visibility. Each adversary is modeled based on their goals, prior knowledge, and capabilities. We then use this model to perform traffic analysis using the Metropolis-Hastings algorithm in conjunction with a Bayesian inference engine. To the best of our knowledge this is the first time such an adversary is explicitly defined and studied, despite this adversary model being championed as more realistic than global passive adversaries in prior work. We highlight that our framework is flexible and able to encompass a broad range of different adversaries. Naturally, our model also captures the classical global passive adversary (GPA) as a special case.

Keywords: Mixnet, Local Adversary, Partial Visibility, Metropolis-Hastings

1 Introduction

The notion of a mixnet was first invented by Chaum [2]. Mixnets were designed specifically to resist an adversary with a global view who can observe all inputs and outputs in the network. Resisting such adversaries is achieved via the mixing strategy where each mix does not output messages immediately upon receiving them. Instead, a mix stores messages for some amount of time before sending them to either another mix or to a recipient. This technique hides correlations between the inputs and outputs which therefore makes mixnets resilient against a Global Passive Adversary (GPA). Currently, there is a large literature that exists that utilizes simulations and empirical evaluation in order to quantify anonymity under such a powerful adversary for generic mixnets [13, 1].

However, adversaries with such full visibility are of questionable practical utility, since in practice even very powerful adversaries may have blind spots in their network coverage [16]. The literature currently lacks methods to model and analyze anonymity properties with respect to an adversary who has partial visibility of the network, such an adversary (such as Iran or Russia surveillance operations) that can only control mixes within their jurisdiction. One challenge in evaluating anonymity under such an adversary is that their advantages in inferring information about messages in a network vary greatly depending on which portions of the network they have visibility over. Deriving, analytically, the probability of an event when considering such an adversary requires conditional probabilities based on the different assumptions of the portions of the network that they can or cannot see. Even a simple network results in an explosion in complexity from the derivation of such probabilities.

To address this problem, we first provide a general and flexible matrix-based mixnet model that captures different mixnet configurations, then we model adversaries based on their goals, priors, and capabilities. By performing this, we retrieve two sets of data: an *Observation* $\mathcal{O}$ and a *Hidden State* $\mathcal{HS}$. The $\mathcal{O}$ is the part of the trace the adversary has visibility over and the $\mathcal{HS}$ is the part of the trace the adversary lacks visibility over. Finally, we use this modeling to perform mixnet traffic analysis under an APV. The main strength of this framework is that it is able to capture a broad range of adversaries. We develop and implement a Bayesian inference engine which takes an APV and a network configuration as input. The inference engine then uses the Metropolis Hastings to estimate the distribution over possible traces.

In summary, our main contributions are as follows:

- A matrix-based mixnet model that is able to capture different mixnet configuration such as different topologies and mixing strategies.
- A model for an *Adversary with Partial Visibility (APV)* that can be captured by the matrix-based mixnet model. This model can be used in conjunction with Bayesian inference techniques to perform traffic analysis attacks. To the best of our knowledge such a general adversary model has not been studied in mixnets before.
- A traffic analysis framework that uses the Metropolis-Hastings algorithm to evaluate anonymity under different APVs.

2 Motivation and Related Work

In this section we present our motivation and related work. First, we discuss Adversaries with Partial Visibility in the context of mixnets. Second, we discuss our decision to choose the Metropolis–Hastings algorithm for traffic analysis.

Adversaries with Partial Visibility: Since Chaum's seminal work on untraceable email [2], there has been considerable research exploring and understanding the design space of mixnets [14, 18, 12, 5]. However, most of this literature assumes a very strong notion of the adversary, called a *Global Passive*

Adversary (GPA), which observes the entire network. In [16], Syverson argues that a GPA is unrealistic and does not reflect the real anonymity of a system. Starting from adversary models that are not reasonable in practice has contributed to the lack of wide-spread adoption of anonymous communication systems. Syverson argues that in security analysis, most adversaries are subject to some constraints on their capabilities such as having a partial view of a network [16]. An adversary's ability to execute a successful attack is dependent on the adversary's available resources and their effectiveness. In other words, the measure of an adversary's ability to succeed must include the resources it will take them to learn enough information to link relations.

In [8], Gallagher et al. have argued for the need for new adversary ontologies that could better model real-world adversaries by outlining the limitations of the adversaries that are commonly used in academic literature, such as Global Passive Adversary. A GPA is unrealistic in real world scenarios where the mixnet system spans multiple administrative domains (e.g., different autonomous systems) in several countries. In fact, considering such an unnecessarily strong adversary may hinder exploring and curtail the development of mixnet designs that can scale better and provide adequate and meaningful privacy in the real world [13].

Traffic Analysis in Mixnets: Multiple variants of traffic analysis attacks in mixnets do exist in the literature [3, 10, 15, 11]. However the focus has been on the edges of mixnets where messages are sent and received and therefore the adversary does traffic analysis on the patterns of these messages in order to correlate for example a sender to a receiver. These attacks do not capture an adversary that is also able to corrupt or monitor a portion of the network in order to de-anonymize the sender and receiver. Traffic analysis by an Adversary with Partial Visibility (APV) over the network is a hard task as it requires consideration of prior knowledge of the adversary as well as the different assumptions for the parts of the network that the adversary does not have visibility over. In [6], the authors argue that at the heart of traffic analysis lies an inference problem. Therefore applying Bayesian techniques provides a sound framework on which to build attacks and algorithms to estimate different quantities. Their main contribution is introducing the application of Bayesian inference to traffic analysis. The Metropolis–Hastings algorithm is a well known method for obtaining a sequence of random samples where direct sampling is difficult [9]. In order to estimate a function of the mapping between input and output messages of the network, they build an inference engine that samples from the distribution of network states that a set of given observations allow. However, this model only models global passive adversaries [6].

One of the key contributions of our work is to extend this bayesian-based framework in conjuction with Metropolis Hastings in order to analyse Adversaries with Partial Visibility. This encompasses a much wider set of adversaries than the Global Passive Adversary considered in prior work [17, 6].

3 Adversary Model

In this paper we provide a method that solves the problem of traffic analysis under an Adversary with Partial Visibility (APV). In a nutshell the problem is presented as follows: The adversary APV has a prior knowledge about the mixnet-based system. An APV observes the network over some time interval t and produces an observation $\mathcal{O}$. Given the prior knowledge on the system model as well as the observation $\mathcal{O}$, the adversary tries to infer the probability of certain events that happened either partially or entirely outside of her visibility during the time t. We call the parts of the network trace that the adversary does not have visibility over a Hidden State $\mathcal{HS}$. Table 1 provides a handy reference for all the notations used in this paper.

One of our objectives for solving this problem is to make the adversary model as general as possible in order to accommodate different types of adversaries. We therefore start by modeling the three major aspects of any adversary: (i) Goal, (ii) Prior and (iii) Capability:

Goal: The adversary chooses a goal. For example an adversary's goal can be to infer the probability of a sender s_i communicating with receiver r_j ($Pr[s_i \rightarrow r_j]$), or correlating a specific input i_i to an output o_j ($Pr[i_i \rightarrow o_j]$). Note that in the first example, the adversary needs to track all messages from s_i to r_j, however in the second example the adversary is only interested in correlating i_i with o_j.

Prior: Depending on the system model, different adversaries can have different sets of prior knowledge. One could imagine a system that have different paths lengths for each message or different constraints on users in choosing the route (for example users in certain geographical locations are more likely to route their messages through nodes in adjacent areas). We use $\mathcal{C}$ for the system constraints. This type of information is encoded in the prior knowledge of the adversary.
To be more formal we assume that an abstract system consists of an Observations $\mathcal{O}$ and a Hidden State $\mathcal{HS}$. We therefore assign a joint probability given the constraints $\mathcal{C}$, $Pr[\mathcal{HS}, \mathcal{O} \mid \mathcal{C}]$. We further elaborate how to incorporate the joint probability in our framework in section 5.

Capability: In order to achieve its goals, the adversary does traffic analysis by making use of its *Compromising* and/or *Monitoring* capabilities. Monitoring the network means that the adversary is monitoring the connections between different physical entities of the network either throughout some portion of the network or throughout the entire network in the case of a GPA. A physical entity in the network can be either a mix μ or a user u. We call this an *inter-entity connection*. An ISP is one example of an adversary that can see inter-entity connections and use this data for traffic analysis. The adversary is also able to *compromise*. Compromising a mix means this adversary has the capability of seeing inside a mix. One real world example of an adversary that can see inner-mix maappings is one that can add mixes to a volunteer-based network [7].

4 A Matrix-based Model of a Mix Network

In this section, we describe our model for a matrix-based mix network. As explained in the previous section, the APV is trying to estimate the probability distribution of the Hidden State $\mathcal{HS}$ given their Observation $\mathcal{O}$ and the prior knowledge about the system constraints $\mathcal{C}$. We therefore need to provide a model for the mixnet system that encompasses the adversary's constraints $\mathcal{C}$ and partitions the message trace (defined next) into an Observation $\mathcal{O}$ and Hidden State $\mathcal{HS}$.

4.1 Trace Graph: A graph derived from the message trace

A trace, which is the full set of messages traversing the network during a time t, is represented by $\mathcal{HS}$ and$\mathcal{O}$. We now describe how to derive a graph $(\mathcal{V}, \mathcal{E})$ from a trace, where $\mathcal{V}$ is the set of vertices in the graph and $\mathcal{E}$ is the set of edges. The purpose of this graph is to split each message path into segments so that we can differentiate between the parts of each path that an adversary can see and the parts that they cannot see. We start by defining a *path*, and then show how to model each path as a graph.

Definition 1 (Path). *A **path**, $P = (s_i, \mu_1, .., \mu_n, r_j) = \{e_1, ..e_{n+2}\}$, is an ordered set of entities, users and mixes, that describes the path of the message from the sender s_i to the receiver r_j (n: length of the message path).*

If P_i is the ith path, then as it goes through mix μ_j, we will denote its input vertex in this mix as $v_{i,I}(\mu_j)$ and its output vertex as $v_{i,O}(\mu_j)$. We will denote the vertex associated with the path's sender as $(v_i(s))$ and the vertex associated with the path's receiver as $(v_i(r))$.

Definition 2 (Path Graph). *Let $P_i = (s, \mu_1, .., \mu_n, r)$ be a path. A path graph is a set of vertices $\mathcal{V}$ and a set of edges $\mathcal{E}$. A vertex represents (i) the sender of the message $(v_i(s))$ of the corresponding path, (ii) the input/output of the message to/from the different mixes, $v_{i,I}(\mu_j)$, $v_{i,O}(\mu_j)$ and (iii) the recipient of the message $(v_i(r))$. An edge is the connections between two vertices of the network, which can be between two entities (either a mix or a user) or between an input vertices and an output vertices of each mix.*

Note that a vertices in the network is not just a physical point of entry of the message. It is rather a representation of time in the graph. For example if a message m arrives to mix μ at $t_1 = 1$ and a message m_2 arrives to μ at time $t_2 = 2$, we assign different vertices in the mix to the different messages. Figure 1b shows a simple example of modeling a portion of the path and its relationship to the mix by vertices and edges.

Definition 3 (Trace). *Let M be a set of messages sent through the mixnet. For message $m \in M$, let P_m be the path of m. A **trace** $\mathcal{TR} = \{P_m : m \in M\}$ is the set of paths of all messages sent through the network during time interval t,*

in other words, a trace is the full set of paths that occur in a network during a time interval t.

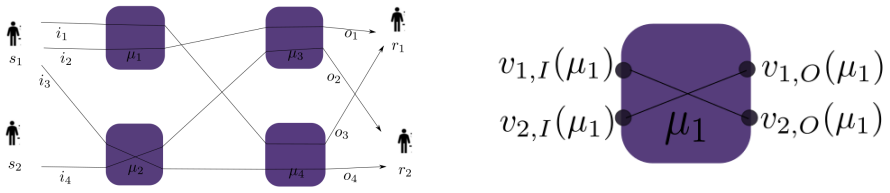
Definition 4 (Trace Graph). *Let M be the set of messages sent across the network during time interval t and let $\mathcal{TR} = \{P_m : m \in M\}$ be a trace of a network in that time interval. Let G_P be the Path graph of path P. The Trace Graph, $G_{TR} = \{G_P : P \in \mathcal{TR}\}$ is the set of Path Graphs associated with paths in $\mathcal{TR}$.*

We model the *Trace Graph* as a set of adjacency matrices.

Definition 5 (Adjacency Matrix). *Let $G = (\mathcal{V}, \mathcal{E})$ be a graph, where $\mathcal{V}$ is the set of vertices and $\mathcal{E}$ is the set of edges. Then an adjacency matrix Mx is a matrix with $|\mathcal{V}|$ columns and $|\mathcal{V}|$ rows, where row i and column i are both labeled by vertex $v_i \in \mathcal{V}$, for $i \in [1, |\mathcal{V}|]$. Element (i, j) of Mx is 1 if the edge $(v_i, v_j) \in \mathcal{E}$ and 0 otherwise.*

Each adjacency matrix will be associated with a subgraph of the Trace Graph, G_{TR}. We identify two types of subgraphs in the Trace Graph. The first type is an inner-mix subgraph, defined by the vertices and edges associated with each mix. We call the adjacency matrix of this subgraph an *inner-mix matrix*. This matrix represents the relationship of the input messages to a mix to the output messages of the same mix.

Definition 6 (Inner-mix matrix). *Let TG be a trace graph, let μ be a mix, and let $\mathcal{V}_I$ be the incoming vertices of messages associated with μ and $\mathcal{V}_O$ be the outgoing vertices of messages associated with μ, as described in Definition 2. For each message going through μ, there is an edge (v, w) between a vertex in $v \in \mathcal{V}_I$ and a vertex $w \in \mathcal{V}_O$, as defined by the path of that message. Let $\mathcal{E}_\mu$ be the set of such edges. Then an inner-mix matrix is the adjacency matrix of the graph $(\mathcal{V}_I \cup \mathcal{V}_O, \mathcal{E}_\mu)$.*



(a) An example of a network trace

(b) Mix μ_1 modeled by vertices and edges.

Fig. 1: An example of network trace $\mathcal{TR}$ and the representation of mix μ_1 by vertices and edges.

We also model *inter-entity connections* as a set of adjacency matrices. An inter-entity matrix is a matrix that represents the connections between entities that have messages going between them.

Definition 7 (Inter-entity matrix). *Let $\mathcal{V}_O$ be the set of outgoing vertices of a subset of the network entities that are able to send messages to subset of entities with vertices $\mathcal{V}_I$. Let $\mathcal{E}_{I,O}$ be the collection of all edges that connect the vertices of $\mathcal{V}_O$ with the vertices $\mathcal{V}_I$. Then an inter-entity matrix is the adjacency matrix of the graph $(V_O \cup V_I, \mathcal{E}_{I,O})$.*

The number of inter-entity matrices in our model vary depending on the network configuration. For example, if we have a free-route network where each mix can send a message to any mix in the network then we can have one inter-entity matrix. However, if the system's topology is stratified then mixes are arranged in a fixed number of layers and can only communicate with mixes in the subsequent layer. In this case we have an inter-entity matrix that connects every two layers (eg $\mathcal{E}_O$ is associated with layer i and $\mathcal{E}_I$ is associated with layer $i+1$).

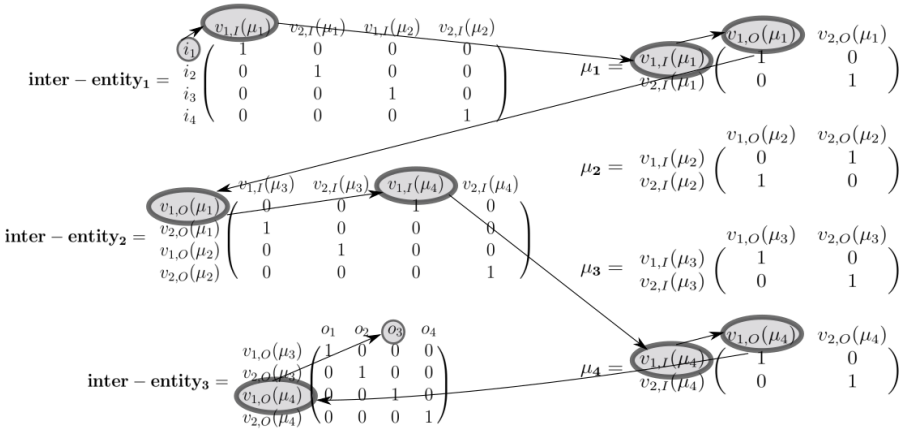


Fig. 2: Modeling the trace in Figure 1a by matrices

4.2 Matrices in two sets

As given in Section 4.1, we model the entire Trace Graph by a list of matrices. There exists two types of matrices (i) inner-mix matrices that represent the "inside" of the mix and (ii) the inter-entities matrices that represent the connections between two sets of entities (either a mix or a user). The motivation behind this modeling is two-fold:

– Using this model we are able to trace any message by simply tracing the elements with value 1 that connect the edges in the different matrices. Figure 2 shows the matrices that are the representation of the trace depicted in Figure 1a. We can see in the Figure 2 how by simply following the connected edges in the matrices we are able to correlate the input i_1 to the output o_3.

– Accommodating the the adversary in our model. The set of matrices represents the full trace of the network, so in order to include the adversary in this model, we need to split the matrices into two sets: matrices that are part of the observation $\mathcal{O}$ and matrices that are part of the Hidden State $\mathcal{HS}$. For example if an adversary APV is only corrupting a mix μ_1, we put the corresponding matrix in the observation and the rest of the matrices in the hidden state. For the inter-entity matrices however, depending on which portions of the network the adversary monitors, the same matrix has to be split to $\mathcal{O}$ and $\mathcal{HS}$. For example if APV controls the connections between μ_1 and μ_3 the *inter-entity*$_2$ in Figure 2 is split in two as shown in Figure 3.

$$\mathcal{O} = \begin{matrix} \\ v_{1,O}(\mu_1) \\ v_{2,O}(\mu_1) \end{matrix} \begin{matrix} \begin{matrix} v_{1,I}(\mu_3) & v_{2,I}(\mu_3) \end{matrix} \\ \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} \end{matrix} \quad \mathcal{HS} = \begin{matrix} \\ v_{1,O}(\mu_2) \\ v_{2,O}(\mu_2) \end{matrix} \begin{matrix} \begin{matrix} v_{1,I}(\mu_4) & v_{2,I}(\mu_4) \end{matrix} \\ \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \end{matrix}$$

Fig. 3: Inter-entity matrix split into $\mathcal{O}$ and $\mathcal{HS}$

We call the *Ground Truth* the trace of messages that actually happens in reality, i.e. the trace that the adversary wants to learn information about. The adversary can see some part of the Ground Truth in its Observation $\mathcal{O}$ and proceeds to conduct the traffic analysis by traversing through the space of the different possible traces in the Hidden State $\mathcal{HS}$.

5 Traffic Analysis of Mixnets

The goal of the APV, having an observation $\mathcal{O}$ and a prior knowledge about the network $\mathcal{C}$, is to estimate the probability distribution over the possible hidden states in order to achieve its goal, for example linking senders to receivers or input messages to output messages. In other words, the APV tries to estimate the target distribution $Pr[\mathcal{HS} \mid \mathcal{O}, \mathcal{C}]$ using the Metropolis–Hastings method.

5.1 Markov Chain Monte Carlo (MCMC)

The Metropolis–Hastings method works by sampling states by conducting a random walk across a state space. In our context, a state is the Hidden State, $\mathcal{HS}$, of the trace. The Metropolis-Hastings algorithm requires that we define a transition function that allows us to propose a new Hidden State ($\mathcal{HS}_p$) for every Hidden State ($\mathcal{HS}_c$) that we encounter in our random walk. Next we compute the following ratio, α_{next_move} in order to decide whether or not to accept the $\mathcal{HS}_p$.

$$\alpha_{next\_move} = \frac{Pr[\mathcal{HS}_c|\mathcal{O}, \mathcal{C}] * Q(\mathcal{HS}_p|\mathcal{HS}_c)}{Pr[\mathcal{HS}_p \mid \mathcal{O}, \mathcal{C}] * Q[\mathcal{HS}_c|\mathcal{HS}_p]} \tag{1}$$

Note that $\mathcal{Q}$ is the transition function that provides the probability of moving from a current state $\mathcal{HS}_c$ to a proposed state $\mathcal{HS}_p$. Further details on how to compute $\mathcal{Q}$ are provided in Algorithm 1. We recall from Section 4 that in the set of matrices $\mathcal{MX}$ associated with a trace $\mathcal{TR}$, a matrix in $\mathcal{MX}$ is either an *inter-entities matrix* or an *inner-mix matrix*. Recall also that we divide this set of matrices into two sets; an Observation $\mathcal{O}$ and a Hidden State $\mathcal{HS}$.

The transition function we use to move from current Hidden State $\mathcal{HS}_c$ to a proposed Hidden State, $\mathcal{HS}_p$ works as follows. We start by choosing a matrix $Mx \in \mathcal{HS}$ at random, and then choosing two columns (which corresponds to vertices in the network) in Mx at random to swap. For inner-mix matrices, this corresponds to the following: let $\mathcal{M}_t$ be a set of messages in a mix with *inner-mix mapping* $F_{\mu_j} : \mathcal{M}_I \rightarrow \mathcal{M}_O$ during time t. Let $v_{1,I}, v_{2,I}(\mu_j) \in \mathcal{M}_I$, and let $v_{1,O}, v_{2,O}(\mu_j) \in \mathcal{M}_O$ such that $F_{\mu_j}(v_{1,I}) = v_{1,O}, F_{\mu_j}(v_{2,I}) = v_{2,O}$. Then swapping two columns in Mx assigns to M_t a new *inner-mix mapping* F'_μ, where $F'_\mu(v_{1,I}(\mu_j)) = v_{2,O}(\mu_j)$ and $F'_\mu(v_{2,I}(\mu_j)) = v_{1,O}(\mu_j)$ and $F'_\mu(x) = F_\mu(x)$ for all x such that $x \notin \{v_{1,I}, v_{2,I}\}$.

For inter-entity connections, the transition function corresponds to the following: Let $\mathcal{ET}_O$ and $\mathcal{ET}_I$ be two sets of entities and let $\mathcal{M}_t$ be a set of messages that are sent from entities in $\mathcal{ET}_O$ to entities in $\mathcal{ET}_I$ during time t. Let $F_e : \mathcal{ET}_O \rightarrow \mathcal{ET}_I$ be the inter-entity connection. Let $\mathcal{V}_O$ be the vertices associated with outgoing packets from $\mathcal{ET}_O$ and let $\mathcal{V}_I$ be the vertices associated with the incoming packets of $\mathcal{ET}_I$. Let $v_{1,O}, v_{2,O} \in \mathcal{V}_O$, and let $v_{1,I}, v_{2,I} \in \mathcal{V}_I$ such that $F_e(v_{1,O}) = v_{1,I}$ and $F_e(v_{2,O}) = v_{2,I}$. Then swapping two columns in Mx assigns to $\mathcal{M}_t$ a new *inter-entity connection* F'_e, where $F'_e(v_{1,O}) = v_{2,I}$, $F'_e(v_{2,O}) = v_{1,I}$ and $F'_e(x) = F_e(x)$ for all x such that $x \notin \{v_{1,O}, v_{2,O}\}$.

5.2 Computing the Probability of a State based on Priors

Depending on the system, the adversary may have prior knowledge on the system, which will be accounted for in the estimation of the target target probability. We model the system by a list of constraints $\mathcal{C}$. Any system can have a variety of constraints, such as paths lengths [17], or user constraints in choosing nodes based on geo-locations, or having a biased distribution for routing (as in the example of Nym [7]).

To consider user constraints, the adversary, having a proposed trace $\mathcal{TR}$, computes $Pr[\mathcal{TR} \mid \mathcal{C}]$ which is the probability of the trace given its prior knowledge on the system model $\mathcal{C}$. As described in Section 4.2, a trace $\mathcal{TR}$ is composed of an Observation $\mathcal{O}$ and $\mathcal{HS}$, therefore:

$$Pr[\mathcal{TR} \mid \mathcal{C}] = Pr[\mathcal{HS}, \mathcal{O} \mid \mathcal{C}] \tag{2}$$

We note however that the adversary is trying to estimate $Pr[\mathcal{HS} \mid \mathcal{O}, \mathcal{C}]$ (Equation 1). We therefore apply Bayes rule:

$$Pr[\mathcal{HS} \mid \mathcal{O}, \mathcal{C}] = \frac{Pr[\mathcal{HS}, \mathcal{O} \mid \mathcal{C}]}{Pr[\mathcal{O} \mid \mathcal{C}]} \tag{3}$$

Thus, we can rewrite $ratio_{next\_move}$ as follows:

$$\begin{aligned}
\alpha_{next_move} &= \frac{Pr[\mathcal{HS}_c \mid \mathcal{O}, \mathcal{C}] * Q(\mathcal{HS}_p \mid \mathcal{HS}_c)}{Pr[\mathcal{HS}_p \mid \mathcal{O}, \mathcal{C}] * Q[\mathcal{HS}_c \mid \mathcal{HS}_p]} \\
&= \frac{Pr[\mathcal{HS}_c, \mathcal{O} \mid \mathcal{C}] / Pr[\mathcal{O} \mid \mathcal{C}] * Q(\mathcal{HS}_p \mid \mathcal{HS}_c)}{Pr[\mathcal{HS}_p, \mathcal{O} \mid \mathcal{C}] / Pr[\mathcal{O} \mid \mathcal{C}] * Q[\mathcal{HS}_c \mid \mathcal{HS}_p]} \\
&= \frac{Pr[\mathcal{HS}_c, \mathcal{O} \mid \mathcal{C}] * Q(\mathcal{HS}_p \mid \mathcal{HS}_c)}{Pr[\mathcal{HS}_p, \mathcal{O} \mid \mathcal{C}] * Q[\mathcal{HS}_c \mid \mathcal{HS}_p]} \\
&= \frac{Pr[\mathcal{TR}_c \mid \mathcal{C}] * Q(\mathcal{TR}_p \mid \mathcal{TR}_c)}{Pr[\mathcal{TR}_p \mid \mathcal{C}] * Q[\mathcal{TR}_c \mid \mathcal{TR}_p]}
\end{aligned} \tag{4}$$

Computing the α_{next_move} is therefore reduced to computing the probabilities of the different traces $\mathcal{TR}$ given the constraints $\mathcal{C}$. Assuming paths selection is independent (we further elaborate on this point in Section 6). Because a trace is a set of message paths, the adversary computes:

$$Pr[\mathcal{TR}|\mathcal{C}] = \prod_{P \in Tr} Pr[P|\mathcal{C}] \tag{5}$$

Based on this derivation, we execute the Metropolis-Hastings algorithm by generating a sequence of different samples (traces) starting from a random trace.

5.3 Metropolis-Hastings Algorithm

We now put everything together to present the concrete algorithm used to gain the target distribution. We describe the algorithm here, and its pseudo-code is included in Algorithm 1.

$Q(\mathcal{TR}_p|\mathcal{TR}_c)$ is the transition function, that is the probability of moving from one state $\mathcal{TR}_c$ to another state $\mathcal{TR}_p$. In our model, this translates to choosing a matrix of $Mx \in \mathcal{MX}$ and two of its columns (c_1, c_2) to permute which yields $\mathcal{TR}_p$. Therefore the transition function is symmetric and equal to $\frac{1}{|\mathcal{MX}|*|\mathcal{V}|*|\mathcal{V}|-1}$, where $|\mathcal{V}|$ is the number of columns in Mx_c. As noted in [4], the values of $Q(\mathcal{TR}_p|\mathcal{TR}_c)$ are not key to the correctness of the sampling, however different transition functions, may speed the convergence to a the stationary distribution.

6 System Model

In this section, we showcase modeling a small example of a mixnet-based system of 9 mixes in order to perform traffic analysis. We choose a stratified topology where the 9 mixes are arranged in 3 layers as shown in Appendix A.1. Each mix is assigned a layer, and messages are sent from layer l_i to layer l_{i+1}. In our experiments, we use a *threshold mix* [2] that buffers messages in the queue until the threshold parameter T is reached. At that point the mix permutes the T messages, and then outputs all messages in a random order. Finally all messages

Algorithm 1: Metropolis–Hastings Algorithm

Result: Computing $Pr[s_i \rightarrow r_j]$
Input: $\mathcal{TR} \leftarrow \{\mathcal{O}, \mathcal{HS}\}$, APV, **burn in,** n_{MH}
Initialize:
$s_i \leftarrow pick\_sender$;
$r_j \leftarrow pick\_receiver$;
$reality \leftarrow$ check$(s_i \rightarrow r_j)$;
$P_t \leftarrow$ compute$(Pr[TR_c])$;
$I \leftarrow 0$;
$n_{samples} \leftarrow 0$;
while $n_{samples} \leq n_{MH}$ **do**
  $MX_c \leftarrow pick\_matrice(HS_c)$;
  $MX_p \leftarrow permute(MX_c)$;
  $HS_p \leftarrow \mathcal{MX}_p = \mathcal{MX} \setminus \{Mx_c\} \cup \{Mx_p\}$;
  $TR_p \leftarrow \mathcal{O}, HS_p$;
  violations $\leftarrow$ check_constraints(TR_p) ;
  if *Not violations* **then**
    $P_{t+1} \leftarrow$ compute$(Pr[TR_p])$;
    $\alpha \leftarrow min\left[1, \frac{P_{t+1}}{P_t}\right]$;
    $r \leftarrow \mathcal{U}(0,1)$;
    if $\alpha \geq r$ **then**
      accept(TR_p);
      $n_{samples}++$;
      if *check*$(s_i \rightarrow r_j)$ **then**
        $I++$;
end
$Pr[s_i \rightarrow r_j] = \frac{I}{n_{samples}}$;
return: $\{Pr[s_i \rightarrow r_j]$, $reality\}$

are source-routed. We emphasize that our model is not limited to this type or size of network . We argue that the matrix-based model is able to accommodate any type of mixes and/or network. However, depending on the type of the network, Equation 5 should be adapted accordingly. Next we show how to compute the probability of a trace in a network of certain constraints $\mathcal{C}$.

6.1 Probability of a Trace

Recall from equation 5, that a trace $\mathcal{TR}$ is a collection of all its paths and therefore the probability of a specific trace is the product of the probabilities of different paths.

$$Pr[\mathcal{TR}|\mathcal{C}] = \prod_{P_i \in \mathcal{TR}} Pr[P_i] \tag{6}$$

In addition, as explained in Section 4, a message path P_i is an ordered set of entities that describe the path of the message from a sender s_i to a receiver r_i. The routing of the message through the different mixes, as well as choosing a

receiver for each message is probabilistic. Let's denote $Pr[\mu]$ the probability of the mix μ being chosen in the path P_i and $Pr[r_i]$ the probability of the receiver r_i is part of P_i, meaning sender s_i chose receiver r_i to send a message to. The probability of the path P_i is then:

$$Pr[P_i|\mathcal{C}] = \left[\prod_{\mu \in P_i} Pr[\mu])\right] Pr[r_i] \tag{7}$$

The above equation holds because we consider source routing, meaning that each user chooses the path of each message among the set of mixes and receivers. We leave other types of routing outside the scope of this work. Putting Equation 6 and Equation 7 together, the probability of a trace is therefore:

$$Pr[\mathcal{TR}|\mathcal{C}] = \prod_{P_i \in \mathcal{TR}} (\prod_{\mu \in P} Pr[\mu]) Pr[r_i] \tag{8}$$

Considering our stratified topology system model of l layers and $N(l_i)$ mixes in each layer l_i, we now proceed to compute the probability of each path. We abuse notation slightly and also consider μ_{j,l_i} as the event that the sender s_i chooses mix $\mu_{j,i}$ as the jth mix in layer l_i, as well as r_i as the event the sender s_i in path i chooses the receiver r_i as the receiver. We first consider that each sender knows all the public parameters of the network such as all the mixes in the network as well as the number of mixes in each layer. The sender then chooses each mix as well as the receiver of each message uniformly.

$$\begin{aligned} Pr[P_i] &= \left[\prod_{i=1}^{l} Pr[\mu_{j,l_i}]\right] Pr[r_i] \\ &= \left[\prod_{i=1}^{l} \frac{1}{N(l_i)}\right] Pr[r_i] \end{aligned} \tag{9}$$

We show how we compute the probability of a trace of a network having other requirements in Appendix A.1.

7 Empirical Evaluation

In this section we first explain how we assess the effectiveness of our model and then show the results of inferring probabilities of trace-related events under two adversaries of different capabilities.

7.1 Determining the Accuracy of our model

Our framework is effective if it reaches the target distribution. To evaluate the effectiveness of our model of an APV we need to assess the closeness of the

samples given by our model to a target distribution based on the ground truth across many rounds of messages in a mixnet.

Recall from Section 5, that we first start by generating the ground truth trace $\mathcal{TR}_{\mathcal{GT}}$ and the adversary specifies the trace related event that they want to infer probability of. We refer to the success or failure of this event in the ground truth as a *goal* and we model it as a predicate function $P : \{\mathcal{TR}_{GT}\} \rightarrow \{0, 1\}$, which takes the ground truth trace$\mathcal{TR}_{GT}$ as input and outputs 1 if the occurrence happens in $\mathcal{TR}_{GT}$ and 0 otherwise. For example, if the goal of the adversary is to infer the probability of sender s_1 sending a message to receiver r_1 during its window of observation, then we associate to this goal a predicate P, where $P(TR) = 1$ if s_1 did send a message to r_1 in the trace TR and 0 otherwise.

Then we translate this ground truth trace into a set of matrices. Depending on what portions of the network the adversary observes, we divide this set of matrices into two sets: Hidden State $\mathcal{HS}$ and $\mathcal{O}$. In order to avoid starting from a high probability state, we do a randomized uniformly chosen number of permutations on the different matrices that are part of $\mathcal{HS}$ and finally we give these two sets of matrices to the adversary as well as the list of constraints.

The adversary then execute the Metropolis–Hastings algorithm on the trace. At the end of one run of the Metropolis–Hastings algorithm, the adversary assign a probability p_{tr} to the event of interest. We refer to the probability, p_{tr}, as a goal and the output of $P(\mathcal{TR}_{GT})$ as an occurrence. We call an occurrence positive when $P(\mathcal{TR}_G) = 1$ and we call an occurrence negative when $P(\mathcal{TR}_G) = 0$.

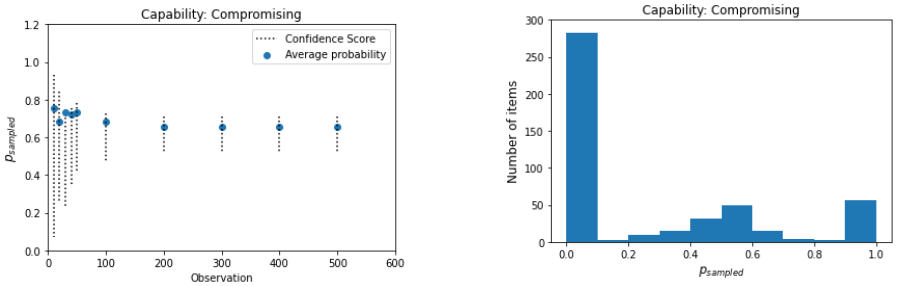
In order to assess the accuracy of our model we need to generalize beyond a single trace. To do this, we run the Metropolis Hastings algorithm on N_{tr} number of traces for the same adversary each time using a different ground truth trace. After each run of the algorithm, we record the probability p_{tr_i} that the adversary assigns a given event to its goal, and we record the occurrence $P(TR_{GT})$. Once we have our full data set $(p_{tr_i}, occurrences)$ tuples, we compare the average inferred probability to the confidence score on the occurrences. We use the Wilson score interval [20], as it has been shown to be the most accurate [19] of binomial confidence intervals. The Wilson Score takes number of successes and number of failures as inputs, and outputs the probability of success along with the 95% confidence interval. We consider our model to be effective if the sampled probability lies within the confidence interval. Therefore, the average ability of the adversary to correctly infer the probability of the specified event across many different events can be taken into account.

7.2 Capability: Compromising

We start by analysing an adversary who has compromising capability. The goal of this adversary is to infer *whether at-least one of the messages of a sender s_i arrived to a receiver r_j*. This adversary tries to link the receiver to senders that route a message through one of the compromised mixes. Therefore we choose an adversary who is compromising one entry mix and one exit mix. We evaluate this adversary for up to 500 observations.

We can see in Figure 4a that when there are only a few observations, the confidence interval is high, however we show that our model eventually converges into the the target distribution where the confidence interval is within tight bounds as the model is guessing with 95% confidence the correct probability. Note however that we are showing the average sampled probability of the different events. The probability of each trace related event may differ per different observation. In Figure 4b we show the histogram of the number of experiments per bin. The idea behind this binning is to analyse beyond the average probability inferred by the adversary for the different observations. After collecting all the sampled probabilities where each probability corresponds to one trace, we store them in different bins depending on their values. Each bin corresponds to an interval of probabilities and contains the number of times the adversary inferred that probability.

We can see that for this adversary, the bin that corresponds to a probability between $[0, 0.1]$ contains the largest number of items, meaning that for most observations the adversary guessed between $[0, 0.1]$ due to lack of information. The second largest bin is the one that corresponds to the events having a probability between $[0.9, 1]$. Those events are likely to be the messages that went through both the compromised mixes (and so the APV had more information), and finally around 25% of the total events the adversary infers a probability between 0.2 and 0.7. This would intuitively match to the case where messages went through a single compromised mix that the APV could watch, but not both.



(a) Accuracy (b) Bins

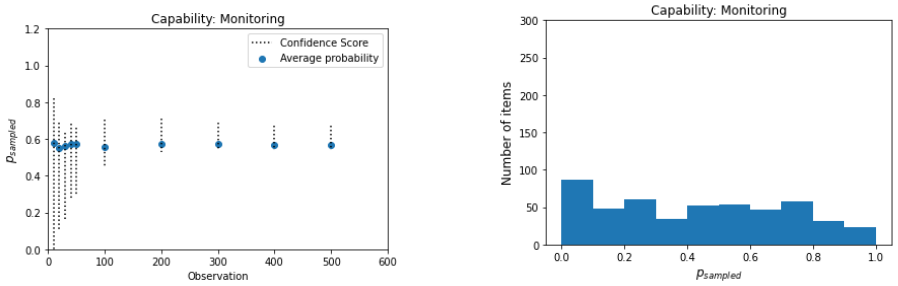
Fig. 4: Capability of APV_1: Compromising 2 mixes

7.3 Capability: Monitoring

Similar to the previous experiment, the goal of this adversary is to infer *whether at-least one of the messages of a sender s_i arrived to a receiver r_j*. The adversary is monitoring the inter-entity matrices, and so the adversary is what is usually called a Global Passive Adversary in mixnet literature. The adversary in this

case is not compromising the mixes, and therefore all permutations of inner-mix mapping are possible.

We can see in Figure 5a that the adversary captured by our model is able to infer probabilities correctly with enough number of observations. As opposed to the APV, the GPA can connect s_i to a receiver r_i with a higher confidence. Figure 5b shows the histogram of the different probabilities estimated by this adversary. We can see that unlike the previous adversary, the number of items inside the different bins is more balanced between probabilities, meaning that this adversary does not sample states with a very high probability compared to other states with low probability. This confirms our understanding with the modeled GPA. In the first adversary we can see that there are traces with high probabilities (a less balanced histogram) that correspond to messages being routed via at least one compromised node. We note however that the average probability inferred about the different events is similar in this scenario even though that the first adversary is more realistic as it compromises only 2 mixes as opposed to the second adversary that monitors all mixes in the network.



(a) Accuracy (b) Bins

Fig. 5: Capability of APV_2: Monitoring all inter-entity matrices (GPA)

We emphasize that the main goal of the paper is not to present the specific results of de-anonymization, but rather to demonstrate the possibility of going beyond the traditional Global Passive Adversary (GPA) model in mixnets. In addition, we show that our model is also able to capture this GPA as a special case.

However, this flexible framework comes with one main drawbacks which is the Metropolis Hastings algorithm is computationally expensive, requiring a large number of state samples to be drawn in order to reach from the target distribution. Additionally, if the adversary is relatively weak, it may be more efficient to rely on the adversary's prior knowledge and analytically infer trace-related events. Nevertheless, for adversaries with a reasonable level of visibility into the network, we consider the computational cost of the algorithm to be acceptable, especially given the lack of alternative methods for studying such adversaries.

7.4 Performance evaluation

Our Metropolis-Hastings sampler is composed by 2363 LOC of Python and run on an Intel(R) Core(TM) i9-9920X with 3.50GHz CPU and 132 GB RAM. As explained in 5.1, in order to compute α_{next_move} we need to compute the probability of every state. Depending on the adversary's knowledge, the network size and the number of messages per observation, the computation time might increase. For our simulated scenarios, it took about 6 for the full analysis. While this may seem expensive, our implementation is not optimised for size, memory usage or running time and we argue that more optimized implementations will outperform it.

8 Conclusion

To the best of our knowledge, this is the first work considering a model of an adversary whose capabilities can be modeled by the partial view of the network. In this paper we present and analyze traffic analysis by Adversaries with Partial Visibility (APV), who have a visibility over some portion of the network. We put forth a mathematical model of a mix network and use this model with the conjunction of Metropolis-Hastings to build a Bayesian inference engine that models how an adversary infer probabilities about trace-specific events. Finally, we analyzed the effectiveness of our model under different simulated adversaries.

The critiques made of previous work on mixnets by Syverson [16] are to a large extent correct, as even powerful nation-state adversaries may have only partial visibility. However, this does not mean that mixnets cannot be analyzed in terms of such adversaries, as our work shows. Our work would allow the more realistic practical engineering of privacy-enhancing technologies, especially as real-world mixnets like Nym are deployed in countries with partial adversaries such as Iran [7].

Nonetheless, the work is limited. While we studied compromised mixes, we assumed the mixnet adversaries that were compromised were honestly following the protocol. Thus, an aspect that needs to be studied is actively malicious mixes as well as users. More importantly further work should systematically study different adversaries, each having different visibility over multiples mix networks with different parameters and design decisions.

Acknowledgement

We thank the anonymous reviewers for their comments and insightful feedback. This research is partially supported by the Research Council KU Leuven under the grant C24/18/049, by CyberSecurity Research Flanders with reference number VR20192203, and by DARPA FA8750-19-C-0502. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of any of the funding agencies.

References

1. Ben Guirat, I., Gosain, D., Diaz, C.: Mixim: Mixnet design decisions and empirical evaluation. In: Proceedings of the 20th Workshop on Workshop on Privacy in the Electronic Society. pp. 33–37 (2021)
2. Chaum, D.: Untraceable electronic mail, return addresses, and digital pseudonyms. Commun. ACM **24**(2), 84–88 (1981). https://doi.org/10.1145/358549.358563, http://doi.acm.org/10.1145/358549.358563
3. Danezis, G.: Statistical disclosure attacks. In: IFIP International Information Security Conference. pp. 421–426. Springer (2003)
4. Danezis, G.: The traffic analysis of continuous-time mixes. In: International Workshop on Privacy Enhancing Technologies. pp. 35–50. Springer (2004)
5. Danezis, G., Dingledine, R., Mathewson, N.: Mixminion: Design of a type iii anonymous remailer protocol. In: Symposium on Security and Privacy, 2003. pp. 2–15. IEEE (2003)
6. Danezis, G., Troncoso, C.: Vida: How to use bayesian inference to de-anonymize persistent communications. In: International Symposium on Privacy Enhancing Technologies Symposium. pp. 56–72. Springer (2009)
7. Diaz, C., Halpin, H., Kiayias, A.: The nym network (2021), https://nymtech.net/nym-whitepaper.pdf
8. Gallagher, K., Barradas, D., Santos, N.: Rethinking realistic adversaries for anonymous communication systems. In: Free and Open Communications on the Internet, FOCI'23. pp. 81–87 (2023), https://petsymposium.org/foci/2023/foci-2023-0015.pdf
9. Hastings, W.K.: Monte Carlo sampling methods using Markov chains and their applications. Oxford University Press (1970)
10. Kesdogan, D., Mölle, D., Richter, S., Rossmanith, P.: Breaking anonymity by learning a unique minimum hitting set. In: International Computer Science Symposium in Russia. pp. 299–309. Springer (2009)
11. Kesdogan, D., Pimenidis, L.: The hitting set attack on anonymity protocols. In: International Workshop on Information Hiding. pp. 326–339. Springer (2004)
12. Kwon, A., Lu, D., Devadas, S.: {XRD}: Scalable messaging system with cryptographic privacy. In: 17th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 20). pp. 759–776 (2020)
13. Piotrowska, A.M.: Studying the anonymity trilemma with a discrete-event mix network simulator. In: Proceedings of the 20th Workshop on Workshop on Privacy in the Electronic Society. pp. 39–44 (2021)
14. Piotrowska, A.M., Hayes, J., Elahi, T., Meiser, S., Danezis, G.: The loopix anonymity system. In: 26th {USENIX} Security Symposium ({USENIX} Security 17). pp. 1199–1216 (2017)
15. Raymond, J.F.: Traffic analysis: Protocols, attacks, design issues, and open problems. In: Designing Privacy Enhancing Technologies. pp. 10–29. Springer (2001)
16. Syverson, P.: Why i'm not an entropist. In: International Workshop on Security Protocols. pp. 213–230. Springer (2009)
17. Troncoso, C., Danezis, G.: The bayesian traffic analysis of mix networks. In: Proceedings of the 16th ACM Conference on Computer and Communications Security. p. 369–379. CCS '09, Association for Computing Machinery, New York, NY, USA (2009). https://doi.org/10.1145/1653662.1653707, https://doi.org/10.1145/1653662.1653707

18. Van Den Hooff, J., Lazar, D., Zaharia, M., Zeldovich, N.: Vuvuzela: Scalable private messaging resistant to traffic analysis. In: Proceedings of the 25th Symposium on Operating Systems Principles. pp. 137–152 (2015)
19. Wallis, S.: Binomial confidence intervals and contingency tests: mathematical fundamentals and the evaluation of alternative methods. Journal of Quantitative Linguistics **20**(3), 178–208 (2013)
20. Wilson, E.B.: Probable inference, the law of succession, and statistical inference. Journal of the American Statistical Association **22**(158), 209–212 (1927)

A System Model: Topology

A.1 Topology

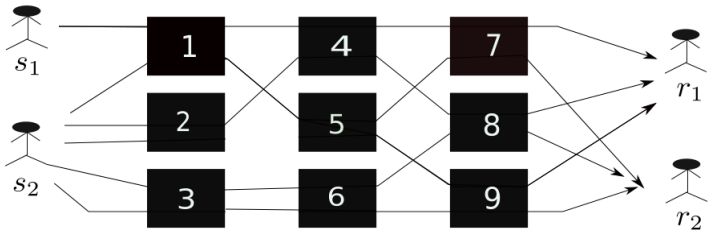


Fig. 6: Mixnet-based system Model

B Probability of a trace: Mixes are not chosen Uniformly

Depending in the systems requirements, mixes can be chosen according to a certain weights assigned to them.We now consider the case of mixes not chosen uniformly, but according to an assigned weight. Mix weights can reflect geographic approximation, bandwidth, trust etc. Let's denote by $w(\mu_{j,l_i}$ the probability of the mix μ_{j,l_i} being chosen among the set of mixes in layer l_i. The probability of each path becomes:

$$Pr[P_i] = \left[\prod_{i=1}^{l} w(\mu_{j_i,i})\right] Pr[r_i] \tag{10}$$

We can also model the case when each sender s_i only knows a subset of the mixes in each layer l_j $(\mathcal{S}(s_i, l_j))$.

$$Pr[P_i|\mathcal{S}(s_i,l_j)] = \left[\prod_{j=1}^{l} Pr[\mu_{j,l_j}|\mathcal{S}(s_i,l_j))]\right] Pr[r_i] \tag{11}$$

To compute $\texttt{Pr}[\mu_{j_i,i}|\mathcal{S}(s_i,l_j)]$, we need to divide the weight of mix $\mu_{j_i,i}$ by the sum of the weights of the all the mixes in layer l_i *that sender* s_i *knows*.

$$Pr[\mu_{j,l_j}|\mathcal{S}(s_i,l_j)] = \frac{w(\mu_{j_i,i})}{w(\mathcal{S}(s_i,l_i))} \tag{12}$$

Therefore:

$$Pr[P_i|\mathcal{S}(s_i,l_j)] = \left[\prod_{j=1}^{l} \frac{w(\mu_{j_i,i})}{w(\mathcal{S}(s_i,l_i))}\right] Pr[r_i] \tag{13}$$

Recipient probability The recipient probability $Pr[r_i]$ captures any prior knowledge of the adversary about the relationship between a specific sender and a specific receiver, for example if the adversary has a prior knowledge about *friendship* graphs of certain senders (i.e sender s_1 is twice as likely to send a message to receiver r_1 than to a receiver r_2).

C Constraints in sampling

Depending on the network configuration as well as the adversary capabilities, some samples states are not possible therefore are thrown away in the Metropolis Hastings algorithm. In order for our model to be accurate we incorporate two categories of these impossible states:

Ghost Messages: In prior work using the Metropolis-Hasting algorithm [4], the authors acknowledge that one of the limitation of their model is assuming that the adversary starts observing the network when the all mixes are empty. We argue however that this is not realistic. Instead the adversary starts observing the network when there are already messages inside the mixes and those message contribute in *hiding* an item of interest. We model this scenario by *ghost messages*. Not to be confused with dummy or cover traffic, ghost messages gm are an output message whose inputs are outside the observation of the adversary and therefore are not items of interests but they do play a role in hiding an item of interest. We therefore add vertices to the subgraph associated with the matrix.

Routing: When an adversary is monitoring or compromising a mix, not only do the have knowledge about the input and output messages of that mix but also the destination and source mixes of those messages. We model this by a list of rules. Any sampled state that is the result of a violations of these rules will be thrown away.

D Table of Notation

Table 1: Summary of notation

Notation	**Interpretation**
Adversary	
$\mathcal{A}$	Adversary
APV	Adversary with Partial Visibility
$\mathcal{O}$	Observation
$\mathcal{HS}$	Hidden State
$\mathcal{C}$	Constraints
Messages	
μ	mix
M_t	Set of messages that are inside μ during t
$\mathcal{M}_I$	Incoming messages to a mix
$\mathcal{M}_O$	Outgoing messages of mix
Matrices	
P_m	Path of the message m
$\mathcal{V}$	Set of Vertices in a path
$\mathcal{E}$	Set of edges in a path
$\mathcal{V}_I$	Set of mix input vertices
$\mathcal{V}_O$	Set of mix output edges
$\mathcal{E}_\mu$	Set of edges in the subgraph associated with the inner-mix mappings of mix μ
$\mathcal{E}_{I,O}$	Set of edges in subgraph associated with inter-entity connections between $\mathcal{ET}_O$ and $\mathcal{ET}_I$
$v_{i,I}(\mu_j)$	ith input edge of jth mix
$v_{i,O}(\mu_j))$	ith output edge of jth mix
$\mathcal{GM}$	Set of ghost messages
Metropolis–Hastings	
$\mathcal{TR}$	Trace
$\mathcal{TR}_p$	Proposed Trace
$\mathcal{TR}_c$	Current Trace
$\mathcal{TR}_G$	Trace of the Ground Truth
Q	Transitioning function