# The Bayesian Traffic Analysis of Mix Networks

Carmela Troncoso
IBBT-K.U.Leuven, ESAT/COSIC,
Kasteelpark Arenberg 10,
B-3001 Leuven-Heverlee, Belgium.
carmela.troncoso@esat.kuleuven.be

George Danezis
Microsoft Research Cambridge,
JJ Thompson Avenue,
Cambridge, UK.
gdane@microsoft.com

## ABSTRACT

This work casts the traffic analysis of anonymity systems, and in particular mix networks, in the context of Bayesian inference. A generative probabilistic model of mix network architectures is presented, that incorporates a number of attack techniques in the traffic analysis literature. We use the model to build an Markov Chain Monte Carlo inference engine, that calculates the probabilities of who is talking to whom given an observation of network traces. We provide a thorough evaluation of its correctness and performance, and confirm that mix networks with realistic parameters are secure. This approach enables us to apply established information theoretic anonymity metrics on complex mix networks, and extract information from anonymised traffic traces optimally.

**Categories and Subject Descriptors:** C.2.0 [Computer-Communication Networks]: General – Security and protection;

**General Terms:** Security, Measurement, Theory.

**Keywords:** Anonymity, Traffic Analysis, Mix Networks, Markov Chain Monte Carlo.

## 1. INTRODUCTION

Relay based anonymous communications were first proposed by David Chaum [2], and have since been the subject of considerable research [5] and deployment [7]. More recently measures of anonymity based on information theory and decision theory were proposed [3, 12, 15, 29, 32] to quantify the security of such systems. Those metrics are based on extracting probability distributions over possible receivers of messages in an anonymity system, subject to constraints on its functioning and the observations of an adversary. Although very popular, these metrics are difficult to apply in the presence of constraints that deployed systems impose, since the exact calculation of the required distributions is an intractable problem (as pointed out by Serjantov [28]).

Our key contribution is a framework to estimate, to an arbitrary accuracy, the distributions necessary for computing a wide variety of anonymity metrics for relay based mix networks. We achieve this by casting the problem of extracting these distributions as a probabilistic inference problem, and solve it using established Bayesian inference frameworks, concretely Markov Chain Monte Carlo (MCMC) sampling.

Our analysis of mix networks incorporates most aspects and attacks previously presented in the literature: constraints on paths length, node selection [29], bridging and fingerprinting attacks [10], social relations of users [16], and erratic user behaviour. For the first time all these aspects on a system are brought under a common framework allowing the adversary to combine them all in the analysis of a system. Further extensions to describe other aspects of mix-networks can also be accommodated. This is the most comprehensive and flexible model of a mix-based anonymity network so far.

The Bayesian traffic analysis techniques presented have two key advantages. First, they allow optimal use of all information when drawing conclusions about who is talking to whom. Second, they provide the analyst with an a-posterior probability over all scenarios of interest, whereas previous attacks only provided the most likely candidate solution. The evaluation of our work focuses on establishing the correctness of those distributions.

Our models of mix networks are far from arbitrary: the parameters and architectures we use model closely the deployed mixmaster and mixminion remailers [7]. The model has also been applied to analyse path creation strategies for low latency, traffic analysis resistant networks [13]. They can be used to assign to messages a correct degree of anonymity, using probabilistic measures [3, 12, 15, 29, 32], which was not possible before. A full account of how to apply those measures can be found in the full version of this paper [11].

The paper is organised as follows: we present a brief overview of Bayesian inference and sampling techniques in Sect. 2; Sect. 3 describes a generic probabilistic model of a mix network, and Sect. 4 shows how to built an inference engine to infer its hidden state; the correctness and accuracy of the inference engine is studied in Sect. 5; and Sect.6 explains how to use the output of the sampler to compute anonymity. Finally we discuss some future directions and conclusions in Sect. 7.

## 2. BAYESIAN INFERENCE AND MONTE CARLO METHODS

Bayesian inference is a branch of statistics with applications to machine learning and estimation. Its key methodology consists of constructing a full probabilistic model of


*CCS'09,* November 9–13, 2009, Chicago, Illinois, USA.

all variables in a system under study. Given observations of some of the variables, the model can be used to extract the probability distributions over the remaining, hidden, variables.

To be more formal lets assume that an abstract system consists of a set of hidden state variables $\mathcal{HS}$ and observations $\mathcal{O}$. We assign to each possible set of these variables a joint probability given a particular model $\mathcal{C}$, $\Pr[\mathcal{HS}, \mathcal{O}|\mathcal{C}]$. By applying Bayes rule we can find the distribution of the hidden state given the observations as:

$$\Pr[\mathcal{HS}|\mathcal{O}, \mathcal{C}] = \frac{\Pr[\mathcal{HS}, \mathcal{O}|\mathcal{C}]}{\sum_{\forall \mathcal{HS}} \Pr[\mathcal{HS}, \mathcal{O}|\mathcal{C}]} = \frac{\Pr[\mathcal{HS}, \mathcal{O}|\mathcal{C}]}{\mathcal{Z}}$$

The joint probability $\Pr[\mathcal{HS}, \mathcal{O}|\mathcal{C}]$ can be decomposed into the equivalent $\Pr[\mathcal{O}|\mathcal{HS}, \mathcal{C}] \cdot \Pr[\mathcal{HS}|\mathcal{C}]$, describing the model and the a-prior distribution over the hidden state. The normalising factor $\mathcal{Z}$ is difficult to compute for large state spaces, and Bayesian techniques do not require it.

There are key advantages in using a Bayesian approach to inference that make it particularly suitable for traffic analysis applications:

- It provides a systematic approach to integrating all information available to an attacker, simply by incorporating all aspects of a system within the probability models [21].
- The problem of traffic analysis is reduced to building a generative model of the system under analysis. Knowing how the system functions is sufficient to encode and perform the attacks, and the inference details are, in theory, easily derived. In practise, computational limitations require carefully crafted models to be able to handle large systems.
- It outputs probability distributions over all possible hidden states, not only the most probable solution as many current traffic analysis methods do.

The last point is an important one: the probability distribution $\Pr[\mathcal{HS}|\mathcal{O}, \mathcal{C}]$ over hidden states given an observation encodes information about all possible states, and the analyst can use it to calculate anonymity metrics [15, 29, 32, 3, 12]. When traffic analysis is used operationally the probability of error of a scenario can be calculated to inform decision making. It is very different to assert that Alice messages Bob, with certainty 99% versus with certainty 5%. Extracting full probability distributions allows us to compute such error estimates directly, without the need for an ad-hoc analysis of false positives and false negatives.

Despite their power Bayesian techniques come at a considerable computational cost. It is not possible to compute directly the distribution $\Pr[\mathcal{HS}|\mathcal{O}, \mathcal{C}]$ due to its complexities and sampling methods have to be used to extract its characteristics: a set of samples $\mathcal{HS}_0, \ldots, \mathcal{HS}_l \sim \Pr[\mathcal{HS}|\mathcal{O}, \mathcal{C}]$ are drawn from the a-posterior distribution, and used to estimate marginal probability distributions of interest.

## 2.1 Metropolis-Hastings Algorithm

The Metropolis-Hastings (MH) algorithm [20] is a Markov Chain Monte Carlo method that can be used to sample from arbitrary distributions. It operates by performing a long random walk on a state space representing the hidden information, using specially crafted transition probabilities that make the walk converge to the target stationary distribution, namely $\Pr[\mathcal{HS}|\mathcal{O}, \mathcal{C}]$. Its operation is often referred to as *simulation*, but we must stress that it is unrelated to simulating the operation of the system under attack.

The MH algorithm's key state is a single instance of the hidden state, called the *current state* and denoted $\mathcal{HS}_j$. Given the current state a another *candidate state* $\mathcal{HS}'$ is selected according to a probability distribution $Q(\mathcal{HS}'|\mathcal{HS}_j)$. A value $\alpha$ is defined as:

$$\alpha = \frac{\Pr[\mathcal{HS}'|\mathcal{O}, \mathcal{C}] \cdot Q(\mathcal{HS}_j|\mathcal{HS}')}{\Pr[\mathcal{HS}_j|\mathcal{O}, \mathcal{C}] \cdot Q(\mathcal{HS}'|\mathcal{HS}_j)}$$

If $\alpha \geq 1$ then the candidate state is accepted as the current state, otherwise it is only accepted with probability $\alpha$. This process is repeated multiple times, and after a certain number of iterations the current state is output as a sample $HS_{j+1}$. More samples can be extracted by repeating this process.

The algorithm is very generic, and can be used to sample from any distribution on any state space, using custom transition probabilities $Q$. It is particularly interesting that the distribution $Q$ used to propose new candidates can be arbitrary without affecting the correctness of the process, as long as both $Q(\mathcal{HS}'|\mathcal{HS}_j) > 0$ and $Q(\mathcal{HS}_j|\mathcal{HS}') > 0$, and the Markov Chain it forms fully connects all hidden states and it is ergodic. Despite the apparent freedom in choosing the distribution $Q$, in practise it must be easy to compute and sample, and be fast mixing to reduce the number of iterations between independent samples. Since the probabilities $\Pr[\mathcal{HS}'|\mathcal{O}, \mathcal{C}]$ and $\Pr[\mathcal{HS}_j|\mathcal{O}, \mathcal{C}]$ need only be known up to a multiplicative constant to calculate $\alpha$, we do not need to know the normalising factor $\mathcal{Z}$.

The other parameters of the MH algorithm, namely the number of iterations necessary per sample, as well as the number of samples are also of some importance.In this work the number of iterations is chosen experimentally to ensure the output samples are statistically independent. The number of MH samples on the other hand impacts on the accuracy of the marginal distributions we estimate, which we can increase by running the sampler longer.

The MH method can be run in parallel on multiple processors, cores or a distributed cluster: all processes output samples that can be aggregated and analysed centrally. Our experiments made use of this property on a multicore two processor machine.

# 3. THE MIX NETWORK MODEL

The first step to perform Bayesian inference is to define a probabilistic model that describes all observations and hidden states of a system. In this section, we present such a model for a set of users sending messages over a mix network to a set of receivers. The model includes traditional aspects of mix networks, e.g. path length constraints, and further incorporates incomplete observations, erratic clients, bridging attacks, and social network information.

We consider an anonymity system formed by $N_{mix}$ threshold mixes [2]. This type of mix achieves unlinkability by collecting $t$ (the threshold) input messages and then outputting them in a random order after a cryptographic transformation. These two actions prevent timing attacks and bitwise linkability respectively (in this work, we assume the cryptography is perfect and leaks no information.) A population of $N_{\text{user}}$ users send messages through these mixes. When

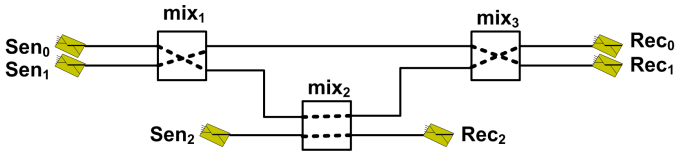



**Figure 1: Observation of the network and Hidden State**

sending a message, a user selects a receiver amongst his set of contacts and a path in the network to route the message. The path is determined by the preferences of the user and a set of constraints $\mathcal{C}$ imposed by the system (e.g., maximum path length, restrictions on the choice of mixes, etc). We denote the sender of an incoming message to the system $i_x$ as $\text{Sen}_x$ and the receiver of an outgoing message from the system $o_y$ as $\text{Rec}_y$.

In order to carry out our analysis we observe the system over a period of time between $T_0$ and $T_{\max}$ (assuming that all mixes are empty at $T_0$.) During this period, $N_{\text{msg}}$ messages travelling through the system are monitored by a passive adversary, generating an *Observation* ($\mathcal{O}$.) This *Observation* is formed by records of communications between the entities (users and mixes) observed by the adversary.

Our goal is to determine the probability of a message entering the network corresponding to each of the messages leaving it given an observation $\mathcal{O}$. This is equivalent to determining the correspondence between inputs and outputs in each of the mixes. We call the collection of the input-output relationships of all mixes the *Hidden State* of the system, and denote it as $\mathcal{HS}$.

Figure 1 depicts an instance of a system where 3 users send 3 messages through a network formed by 3 threshold mixes with threshold $t = 2$. In this setting a passive observer can monitor the following events ($\alpha \to \beta$ denotes entity $\alpha$ sending a message to entity $\beta$) and construct an observation $\mathcal{O}$ with them:

$$\begin{aligned}\mathcal{O} = \{ \quad & \text{Sen}_0 \to \text{mix}_1\ , \quad \text{mix}_1 \to \text{mix}_3\ , \quad \text{mix}_2 \to \text{Rec}_2\ , \\ & \text{Sen}_1 \to \text{mix}_1\ , \quad \text{mix}_3 \to \text{mix}_2\ , \quad \text{mix}_3 \to \text{Rec}_0\ , \\ & \text{Sen}_2 \to \text{mix}_2\ , \quad \text{mix}_3 \to \text{mix}_2\ , \quad \text{mix}_3 \to \text{Rec}_1\ \}\end{aligned}$$

These events are represented with solid lines in Fig. 1. A possible $\mathcal{HS}$ (correspondences between incoming and outgoing messages at all mixes) for this instance is represented with dashed lines.

Given an observation and a hidden state we define a path $P_x$ for each of the messages $i_x$ entering the network, which represents its trajectory through the system. A path consists of a series of observed events that are linked by the relations stated in the Hidden State. In the example, message $i_1$ follows the path $P_1 = \{\text{Sen}_1 \to \text{mix}_1, \text{mix}_1 \to \text{mix}_3, \text{mix}_3 \to \text{Rec}_1\}$. We note that a set of paths $\mathcal{P} = \{P_x,\ x = 1, \ldots, N_{\text{msg}}\}$ defines uniquely an observation and a hidden state. Hence, given a set of constraints $\mathcal{C}$, their probability should be strictly equal, i.e. $\Pr[\mathcal{P}|\mathcal{C}] = \Pr[\mathcal{O}, \mathcal{HS}|\mathcal{C}]$. By applying Bayes theorem we can relate the probability of a hidden state (that we are trying to infer) to the observations and the paths that it forms as $\Pr[\mathcal{HS}|\mathcal{O}, \mathcal{C}] = \Pr[\mathcal{P}|\mathcal{C}]/\mathcal{Z}$ where $\mathcal{Z}$ is a normalising constant.

Hence, we can say that sampling hidden states $\Pr[\mathcal{HS}|\mathcal{O}, \mathcal{C}]$ is equivalent to sampling paths $\Pr[\mathcal{P}|\mathcal{C}]$ using Bayesian inference techniques. In the next sections we present a probability model of paths under different system-based and user-based constraints. Ultimately, we use the Metropolis-Hastings method to sample from that model.

## 3.1 Basic Constraints

First, we present our model for *basic* constraints concerning the user's choice of mixes to relay messages and the length of the path.

We assume that the system allows the user to choose paths of length $L_x, L_x = L_{\min}, \ldots, L_{\max}$. We consider that the user selects this length uniformly at random amongst the possible values. There is nothing special about the uniform distribution of path lengths, and an arbitrary distribution can be used instead. The probability of path $P_x$ being of length $l$ is:

$$\Pr[L_x = l|\mathcal{C}] = \frac{1}{L_{\max} - L_{\min} + 1}\,.$$

Once the length is determined, the user has to choose the mixes on the path. We consider any sequence of mixes of the chosen length as equally likely, with the only condition that mixes have to be distinct. The possible ways in which the $l$ mixes forming a path can be chosen is given by the permutations of length $l$ out of the $N_{\text{mix}}$ mixes forming the system. Thus, the probability of choosing a valid sequence of mixes of length $L_x$ is:

$$\Pr[M_x|L_x = l, \mathcal{C}] = \frac{(N_{\text{mix}} - l)!}{N_{\text{mix}}!}\,.$$

Assuming that the choice of the length of a path and the choice of mixes belonging to it are independent, the probability of selecting a path $P_x$ is:

$$\Pr[P_x|\mathcal{C}] = \Pr[L_x = l|\mathcal{C}] \cdot \Pr[M_x|L_x = l, \mathcal{C}] \cdot I_{\text{set}}(P_x)\,, \quad (1)$$

where the last element represents an indicator of the choice of mixes being a set or a multiset. This indicator takes value 1 when all mixes in the path are different, 0 otherwise.

Since the observation is limited in time, it may be the case that some messages enter the network but never leave it. This happens when messages enter mixes that do not receive enough inputs during the time of observation in order to flush, and thus stay in those mixes at the end. For these messages, it is not possible to derive the choices of the user in terms of path length and mixes, as we can only observe part of the path. Such an example is the observation shown in Fig. 2, representing an instance of a network formed by threshold mixes ($t = 2$) in which users can choose paths of length $L_x \in [2, 3]$. The message sent by $\text{Sen}_2$ arrives at $\text{mix}_4$, but is never forwarded to any other mix or to its recipient since no more messages are received by this mix. At this point, an adversary cannot assume $\text{Sen}_2$ chose $L_2 = 2$, and must consider also the possibility that the choice was $L_2 = 3$.

The probability of a path $P_x$ ending in an unflushed mix is:

$$\Pr[P_{x,\text{unf}}|\mathcal{C}] = \sum_{l=L_{\text{unf}}}^{L_{\max}} \Pr[L_x = l|\mathcal{C}] \cdot \Pr[M_x|L_x = l, \mathcal{C}]\,,$$

where $L_{\text{unf}} = \min(L_{\min}, L_{\text{obs}})$, and $L_{\text{obs}}$ is the observed length of the path from the sender until the mix that has not flushed.

As we have shown, the probability of a hidden state is proportional to the joint probability of the paths chosen by

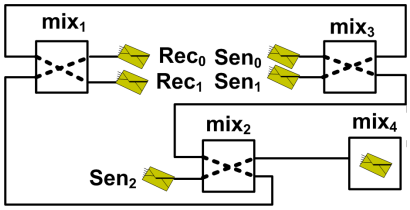



**Figure 2: Example where some mixes do not flush**

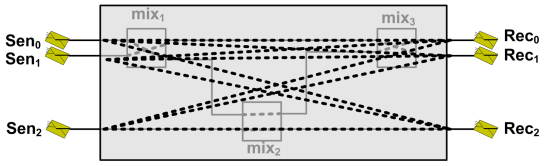



**Figure 3: Black box abstraction of the system**

the users. Assuming users decide independently about the routing of their messages:

$$\Pr[\mathcal{HS}|\mathcal{O},\mathcal{C}] \propto \Pr[\mathcal{P}|\mathcal{C}] = \prod_{x=1}^{N_{\text{msg}}} \Pr[P_x|\mathcal{C}]\,. \tag{2}$$

Further, one can abstract the system as a black box such that there exist a one-to-one relationship amongst incoming and outgoing messages (Fig. 3 depicts an example of this abstraction for the network in Fig 1.) In other words the messages at the exit of the black box must be a permutation of the messages at the entrance [33]. The number of permutations of $N_{\text{msg}}$ messages is $N_{\text{msg}}!$. Without any a-priori information, the probability of the real permutation being any of them is: $1/N_{\text{msg}}!$. This information can be integrated in the computation of the probability of a set of paths:

$$\Pr[\mathcal{P}|\mathcal{C}] = \prod_{x=1}^{N_{\text{msg}}} \Pr[P_x|\mathcal{C}] \cdot \frac{1}{N_{\text{msg}}!}\,. \tag{3}$$

## 3.2 Advanced Constraints

In this section we present our modeling of *advanced* constraints which account for additional knowledge of the adversary about the users' behaviour. The constraints described here can be selectively combined to refine the probabilistic model of the system, resulting in more accurate attacks.

### 3.2.1 Bridging & mix preferences

Bridging attacks were proposed by Danezis and Syverson in [10]. These attacks exploit the fact that users of a large anonymity network might not know all the mixes present in the system. In this case it is possible to "bridge" honest mixes considering the knowledge (or ignorance) about subsequent mixes in a path that the originator of the communication has. For example, given a message sent through a honest mix the path followed by this message can be "bridged" either if (i) there is only one outgoing mix known by its sender, or (ii) if there is only one outgoing mix that is not known by all the senders of the other messages present in the round.

Bridging attacks can be incorporated in our model through the definition of a new indicator variable $I_{\text{bridge}}(P_x)$ associated with each path. This variable takes the value 1 if all mixes in a given path $P_x$ are known to the initiator of the path, and is set to 0 otherwise. We can easily integrate bridging in Eq. 1:

$$\Pr[P_x|\mathcal{C}] = \Pr[L_x = l|\mathcal{C}]\cdot\Pr[M_x|L_x = l,\mathcal{C}]\cdot I_{\text{set}}(P_x)\cdot I_{\text{bridge}}(P_x)\,.$$

This probability can in turn be used in Eq. 3 to obtain the probability of a set of paths $\mathcal{P}$.

A probabilistic version of bridging can also be incorporated into the model, moving beyond the possibilistic bridging attacks described in [10]. Detailed knowledge of the attacker as to which client knows which server, as well as their probability of choosing it, can be used to build probability distributions over the paths $\Pr[P_x|\text{Sender}(P_x),\mathcal{C}]$. Such distributions can represent the knowledge of each sender about the mix network infrastructure, but also any preferences they might have about the choice of mixes. The use of guard nodes [34] in Tor [17] can be modelled in this manner.

### 3.2.2 Non-compliant Clients

Our model so far assumes that all clients make routing decisions according to the standard parameters of the system. This is overwhelmingly the case, since most users will be downloading client software that builds paths for them in a particular, and known fashion. We call those clients and the paths they create *compliant*. For example, the Tor [17] standard client will choose paths of length three as well as distinct onion routers. Furthermore the first router will be a "guard" [34] node. However, some users may modify the configuration of their client to chose paths in a different fashion.

Paths built by these *non-compliant* clients have different probabilities from what our model has assumed so far. We are very liberal with those paths, and make as few assumptions as possible about them. Non-compliant clients may select shorter or longer path lengths than usual in the system, i.e., $L_{\overline{cp}} = L_{\min_{\overline{cp}}},\ldots,L_{\max_{\overline{cp}}}$ with $L_{\min_{\overline{cp}}} \neq L_{\min}$ and $L_{\max_{\overline{cp}}} \neq L_{\max}$. Furthermore, they may use a multiset of mixes to route their messages. The probability of their path is:

$$\Pr[P_x|\overline{\mathcal{C}}] = \frac{1}{L_{\max_{\overline{cp}}} - L_{\min_{\overline{cp}}} + 1} \cdot \frac{1}{N_{\text{mix}}^{l}}$$

For this probability, we have arbitrarily chosen a uniform distribution for the length of the paths, but the model allows to consider any other distribution instead. We indicate with $\overline{\mathcal{C}}$ that the path has been constructed by a non-compliant user. Finally, note that the indicator variable $I_{\text{set}}(P_x)$ enforcing the need for selecting distinct nodes on the path has disappeared from the equation with respect to Eq. 1.

If bridging information is available to the adversary, the indicator $I_{\text{bridge}}(P_x)$ can still be used in the formula to account for user's partial knowledge of the network and increase the accuracy of the attack. This attack is still applicable to non-compliant users, as the fact that they choose routing paths based on their own criterion does not affect the mixes they know.

In order to account for this type of clients, we assume that individual users are non-compliant with probability $p_{\overline{cp}}$. If non-compliant clients are present in the network, we calculate the joint probability of all paths assuming that each user is compliant or not independently, and then assigning a probability to their path accordingly. We denote $P_{cp}$ and $P_{\overline{cp}}$

the set of paths originated by compliant and non-compliant users respectively. We extend the probability model from Sect. 3.1 and derive:

$$\Pr[\mathcal{P}|\mathcal{C}] = \Pr[\pi_i] \cdot \left[ \prod_{P_i \in P_{\overline{cp}}} p_{\overline{cp}} \Pr(P_i|\overline{\mathcal{C}}) \right] \cdot \left[ \prod_{P_j \in P_{cp}} (1 - p_{\overline{cp}}) \Pr(P_j|\mathcal{C}) \right],$$

### 3.2.3 *Integrating Social Network Information*

A number of attacks, starting by Kesdogan *et al* in [22, 1], and further studied in [4, 6, 9, 23, 25, 33], show that adversaries can sometimes extract general profiles of the "friends" of users. These social profiles can then be integrated in the traffic analysis process to narrow down who the receiver of each sent message is[11].

Let us assume that each sender $\text{Sen}_x$ can be associated with a sending profile, i.e., a probability distribution where each element $\Pr[\text{Sen}_x \to \text{Rec}_y]$ expresses the probability of sender $\text{Sen}_x$ choosing $\text{Rec}_y$ as the recipient of a message. We can include this "profile" on the path probability calculation. In this case, Eq. 1 becomes:

$$\Pr[P_x|\mathcal{C}] = \Pr[L_x = l|\mathcal{C}] \cdot \Pr[M_x|L_x = l, \mathcal{C}] \cdot I_{\text{set}}(P_x) \cdot \Pr[\text{Sen}_x \to \text{Rec}_y],$$

$\text{Sen}_x$ being the originator of the path $P_x$ and $\text{Rec}_y$ the recipient of her message. This probability is in turn used in Eq. 2 to calculate the probability of a hidden state. Note that Eq. 3 does not apply anymore as the permutation information is now included in the computation of the probability of a path. Of course, further restrictions as bridging information or considering some senders as non-compliant can be integrated in this probability computation.

## 4. A MARKOV CHAIN MONTE CARLO SAMPLER FOR MIX NETWORKS

Given an observation $\mathcal{O}$ of some messages' flow in an anonymity network and some knowledge about its functioning and its users' behaviour $\mathcal{C}$, traffic analysis aims to uncover the relation between senders and receivers, or equivalently to find the links between incoming and outgoing messages. This comes down to obtaining an a-posterior distribution $\Pr[\mathcal{HS}|\mathcal{O}, \mathcal{C}]$ of hidden states $\mathcal{HS}$ given an observation $\mathcal{O}$ and a set of constraints $\mathcal{C}$.

However, enumerating $\Pr[\mathcal{HS}|\mathcal{O}, \mathcal{C}]$ for all $\mathcal{HS}$ is computationally unfeasible, due to the very large number of possible hidden states. Instead we have shown in Sect. 3 that we can sample states $\mathcal{HS} \sim \Pr[\mathcal{P}|\mathcal{C}]$. These samples are then used to infer the distributions that describe events of interest in the system. For instance, it is easy to estimate the probability $\Pr[i_x \to o_y|\mathcal{O}, \mathcal{C}]$ of an incoming message $i_x$ corresponding to any of the outgoing messages $o_y$ as:

$$\Pr[i_x \to o_y|\mathcal{O}, \mathcal{C}] \approx \frac{\sum_{j \in N_{\text{MH}}} I_{i_x \to o_x}(\mathcal{HS}_j)}{N_{\text{MH}}},$$

where $I_{i_x \to o_x}(\mathcal{HS}_j)$ is an indicator variable expressing if messages $i_x$ and $o_y$ are linked in hidden state $\mathcal{HS}_j$, and $N_{\text{MH}}$ is the number of samples $\mathcal{HS} \sim \Pr[\mathcal{P}|\mathcal{C}]$ available to the adversary. The same process can be used to estimate the sending profile $\Pr[\text{Sen}_x \to \text{Rec}_x|\mathcal{O}, \mathcal{C}]$ of a given user by substituting the indicator variable in the previous equation by $I_{\text{Sen}_x \to \text{Rec}_x}(\mathcal{HS}_j)$.

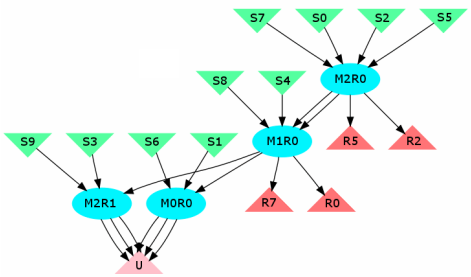



**Figure 4: Observation of a network where 10 messages are sent to 3 mixes of threshold $t = 4$**

We present a Metropolis-Hastings (MH) sampler for $\Pr[\mathcal{P}|\mathcal{C}]$ following the probability mix network model described in Sect. 3. For the sake of simplicity in the remainder of the section we omit the conditioning to the observation $\mathcal{O}$ and the constraints $\mathcal{C}$ in all probabilities (e.g., we write $\Pr[i_x \to o_y]$ when we refer to $\Pr[i_x \to o_y|\mathcal{O}, \mathcal{C}]$) unless stated differently.

### 4.1 Metropolis-Hastings Sampler

Let us consider an anonymity network where users behave as described in Sect. 3. An instance of such a network where 10 messages are sent through 3 mixes of threshold $t = 4$ can be seen in Fig. 4. In this figure, senders are represented as triangles and labeled "S$n$", $n$ being their identity. Likewise for receivers, represented as triangles labelled "R$n$". The triangle labelled as "U" represents *Unknown*, a fake receiver necessary to model the messages that stay in mixes that have not flushed at the end of the observation period. Finally, mixes are represented as ovals, and labelled as "M$m$R$r$", where $m$ expresses the identity of the mix and $r$ the round of flushing. (A non-toy example of a trace can also be seen in Figure 9 in the appendix.)

Note that, although we consider that the network consists of three mixes (M0, M1 and M2), messages seem to be sent to 4 different mixes (M0R0, M1R0, M2R0 and M2R1.) This reflects the fact that messages sent to the same mix in separate rounds do not mix with each other. Let us call the latter series of mixes "virtual mixes" and denote the set they form as vmixes (in the example vmixes = {M0R0, M1R0, M2R0, M2R1})

We define a hidden state as a set of internal connections between inputs and outputs in the virtual mixes, such that an input corresponds to one, and only one, output. The aim of the sampler is to provide hidden state samples, according to the actual probability distribution over all possible hidden states. We compute the probability of a hidden state $\Pr[\mathcal{HS}|\mathcal{O}, \mathcal{C}] \propto \Pr[\mathcal{P}|\mathcal{C}]$ following the model presented in Sect. 3 with both basic and advanced constraints. For simplicity, we denote this probability as $\Pr[\mathcal{HS}]$ in the remainder of the section.

We now explain how to ensure that the random walk performed by the Metropolis-Hastings algorithm actually provides samples from the target distribution $\Pr[HS]$. Let us start by considering only basic constraints (see Sect. 3.1) on the system. We select an arbitrary initial state and use different transitions $Q$ to propose new candidate states for the random walk. When only basic constraints are considered


**Figure 5: $Q_{\mathbf{swap}}$ transition operation on the second and third links of a mix**

we define two transitions:

- $Q_{\mathbf{none}}$: this transition does not change the current state (i.e., the current state is the candidate for next state in the walk),
- $Q_{\mathbf{swap}}$: this transition swaps two internal connections in a virtual mix (See Fig. 5.)

Given a state $\mathcal{HS}_j$ and a transition $Q$ that leads to the candidate state $\mathcal{HS}'$, we decide whether $\mathcal{HS}'$ is a suitable next state for the walk by computing $\alpha$:

$$\alpha = \frac{\Pr[\mathcal{HS}'] \cdot Q(\mathcal{HS}_j|\mathcal{HS}')}{\Pr[\mathcal{HS}_j] \cdot Q(\mathcal{HS}'|\mathcal{HS}_j)} .$$

The new state $\mathcal{HS}'$ is accepted with probability 1 if $\alpha \geq 1$ or with probability $\alpha$ otherwise, as the Metropolis-Hastings algorithm dictates (Sect. 2.1.)

$Q(\mathcal{HS}'|\mathcal{HS}_j)$ (and conversely $Q(\mathcal{HS}_j|\mathcal{HS}')$) is the probability of selecting state $\mathcal{HS}'$ as candidate given that the previous state was $\mathcal{HS}_j$. It depends on the transition Q selected and the probability of selecting this transformation ($\Pr[Q_x]$, $x = \text{none}, \text{swap}$):

$$Q(\mathcal{HS}'|\mathcal{HS}_j) = \begin{cases} \Pr[Q_{\text{none}}] & \text{if } Q_{\text{none}} \\ \Pr[Q_{\text{swap}}] \cdot \frac{1}{|\text{vmixes}|} \cdot \frac{1}{\text{t}} \frac{1}{t-1} & \text{if } Q_{\text{swap}} \end{cases}$$

When taking into account non-compliant clients, the hidden states are not anymore uniquely defined by the set of internal connections in the virtual mixes "present" in the observation $\mathcal{O}$. In this case a client $\text{Sen}_x$ can be compliant or non-compliant ($\text{Sen}_{x,cp}$ or $\text{Sen}_{x,\overline{cp}}$, respectively) resulting in a different probability for the path $P_x$ it initiates, and hence leading to different hidden state probabilities $\Pr[\mathcal{HS}]$. We augment the hidden state to include the internal connections in the virtual mixes, as well as path labels indicating whether the paths are compliant with the system or not.

In this augmented model the random walk $Q$ must modulate the path's labels as compliant or not. Thus, each time a path is altered by a swap operation from the current state $\mathcal{HS}_j$ to create a candidate state $\mathcal{HS}'$, we consider whether to change its sender's label. At iteration $j+1$, we change sender $\text{Sen}_x$'s label depending on the label it had in the previous iteration $j$ (i.e., in hidden state $\mathcal{HS}_j$) and on whether the new path in the candidate state $\mathcal{HS}'$ complies with the system standard parameters or not. We define the probability of a label being changed as:

$$p_{\text{flip}}(a,b) = \Pr[\text{Sen}_{x,b} \text{ in } \mathcal{HS}'|\text{Sen}_{x,a} \text{ in } \mathcal{HS}_j] , \; a,b = \{cp, \overline{cp}\} .$$

$$Q(\mathcal{HS}'|\mathcal{HS}_j) = \begin{cases} \Pr[Q_{\text{none}}] & \text{if } Q_{\text{none}} \\ \Pr[Q_{\text{swap}}] \cdot \frac{1}{\text{vmix}_{\text{max}}} \cdot \frac{1}{\text{t}} \frac{1}{t-1} \cdot Q_{\text{flip}} & \text{if } Q_{\text{swap}} \end{cases}$$

where $Q_{\text{flip}}$ is computed on the paths participating in the swap as:

$$Q_{\text{flip}} = \prod_{\substack{\text{Lab}_x \text{ in } \mathcal{HS}' \\ \neq \text{Lab}_x \text{ in } \mathcal{HS}_j}} p_{\text{flip}} \cdot \prod_{\substack{\text{Lab}_x \text{ in } \mathcal{HS}' \\ = \text{Lab}_x \text{ in } \mathcal{HS}_j}} (1 - p_{\text{flip}}) .$$

Some input messages have "deterministic paths", meaning that their paths are uniquely determined. This is the case, for example, when a message immediately enters a mix that is never flushed. As a result, the label of the path would be never changed and some possible hidden states would never be visited by the random walk. To avoid these cases and ensure that the sampler explores the full state space we define a third type of transition:

- $Q_{\mathbf{det}}$: this transition modifies the compliant status of the sender of one of the $N_{\text{det}}$ deterministic paths present in the network. If no clients are deemed to be non-compliant or no deterministic paths exist, this transition is never applied ($\Pr[Q_{\text{det}}] = 0$.)

Given all these transitions, we compute $Q(\mathcal{HS}'|\mathcal{HS}_j)$ as:

$$Q(\mathcal{HS}'|\mathcal{HS}_j) = \begin{cases} \Pr[Q_{\text{none}}] & \text{if } Q_{\text{none}} \\ \Pr[Q_{\text{swap}}] \cdot \frac{1}{\text{vmix}_{max}} \cdot \frac{1}{\text{t}} \frac{1}{t-1} \cdot Q_{\text{flip}} & \text{if } Q_{\text{swap}} \\ \Pr[Q_{\text{det}}] \cdot \frac{1}{N_{\text{det}}} & \text{if } Q_{\text{det}} \end{cases}$$

# 5. EVALUATION

The aim of our evaluation is to ensure that the inferences drawn from the Metropolis-Hastings samples are "correct". Correctness means that the a-posterior distributions returned represent indeed the probabilities of paths, and correspondences between senders and receivers, in the system.

We evaluate the inference engine with small (3 mixes) and large (5 to 10 mixes) networks. For these networks, we create different observations inserting $N_{\text{msg}}$ messages ($N_{\text{msg}} \in \{10, 50, 100, 1000\}$) from users that choose paths of length between $L_{\text{min}} = 1$ and $L_{\text{max}} = 3$ and select the mixes belonging to these paths uniformly at random. In some of the experiments, we consider the users to be non-compliant with probability $p_{\overline{cp}} = 0.1$.

## 5.1 Metropolis-Hastings parameters

The sampler parameters are important to ensure that the samples returned are from the desired distribution $\Pr[\mathcal{HS}]$.

The number of iterations $\iota$ between samples must guarantee they are independent. There is no straightforward procedure to obtain the optimal value for this parameter and we have to estimate it. We consider $\iota$ to be large enough when the second order statistics of the marginal distributions $\Pr[i_x \rightarrow o_y]$ (respectively $\Pr[\text{Sen}_x \rightarrow \text{Rec}_y]$) are the same as the first order statistics. Informally we want to ensure that the probability that an input $i_x$ corresponds to an output $o_y$ at sample $j$ is independent of the output of $i_x$ at sample $j-1$. Formally, the property we are looking for is:

$$\Pr[i_x \rightarrow o_y \text{ in } \mathcal{HS}_j | i_x \rightarrow o_h \text{ in } \mathcal{HS}_{j-1}] = \Pr[i_x \rightarrow o_y \text{ in } \mathcal{HS}_j] , \quad (4)$$

for some $h$. We experimentally test different values of $\iota$ for independence to determine it.

The higher the number of samples $N_{\text{MH}}$ extracted, the better the estimate of the a-posterior distributions at the cost of more computation. We chose the number of samples for our problems based on the order of magnitude of the a-posterior probabilities we expect to infer.

When adapting our experiments to consider non-compliant clients, we need to chose a value for the parameters $p_{\overline{cp}}$ and $p_{\text{flip}}(a,b)$, the average percentage of non-compliant clients in the network. We decided to assign $p_{\overline{cp}} = 0.1$ such that the percentage of non-compliant clients using the network is

| | Parameter | Value | | | |
|---|---|---|---|---|---|
| Network parameters | $N_{\text{msg}}$ | 10 | 50 | 100 | 1000 |
| | $N_{\text{mix}}$ | 3 | 3 | 10 | 10 |
| | t | 3 | 3 | 20 | 20 |
| | $[L_{\min}, L_{\max}]$ | [1,3] | | | |
| Advanced constraints | $p_{\overline{cp}}$ | 0.1 | | | |
| | $p_{\text{flip}}(\overline{cp}, cp)$ | 0.9 | | | |
| | $p_{\text{flip}}(\overline{cp}, \overline{cp})$ | 0.01 | | | |
| | $p_{\text{flip}}(cp, cp)$ | 0.02 | | | |
| | $p_{\text{flip}}(cp, \overline{cp})$ | 0.3 | | | |
| | $[L_{\min_{\overline{cp}}}, L_{\max_{\overline{cp}}}]$ | [1,32] | | | |
| Sampler parameters | $\iota$ | 6011 | 6011 | 7011 | 7011 |
| | burn-in | 8011 | | | |
| | $N_{\text{MH}}$ | 500 | 500 | 500 | 500 |

**Table 1: Parameters of the Metropolis-Hastings sampler implementation**

small (as expected in a real network) but their presence in the network is non-negligible to impact the analysis.

The probability $p_{\text{flip}}(a, b)$ is not crucial for the correctness of the sampler, but is important to the speed of mixing. The values we used in our experiments were chosen empirically to ensure fast mixing. A study of optimal values for $p_{\text{flip}}(a, b)$ given $p_{\overline{cp}}$[1] is left as subject of future research.

The values for the parameters used in our experiments are summarised in Table 1. We chose the network parameters to produce observations that are interesting. Had we always considered a realistic mix network, with at least $N_{\text{mix}} = 10$ with threshold $t = 10$, and few messages (10 or 50), we would run the risk of many mixes not flushing and therefore not observing any flow of messages.

## 5.2 Evaluation methodology

For a given observation, we collect $N_{\text{MH}}$ samples from $\Pr[\mathcal{HS}]$ using the Metropolis-Hastings algorithm with the transitions $Q$ described. Using these samples we estimate the marginal probability distributions $\Pr[i_x \to o_y]$ and $\Pr[\text{Sen}_x \to \text{Rec}_y]$, linking input messages to output messages and senders to receivers respectively (as demonstrated in [19] there can be a substantial difference between them).

Let us call each of the samples obtained in the MH simulation $\mathcal{HS}_j$, $j \in \{1, \ldots, N_{\text{MH}}\}$. The result of our basic experiment is a point estimate of $\Pr[i_x \to o_y]$ (respectively, $\Pr[\text{Sen}_x \to \text{Rec}_y]$) for each of the messages $i_x$ entering the network:

$$\Pr[i_x \to o_y] = \frac{\sum_{j \in N_{\text{MH}}} I_{i_x \to o_y}(\mathcal{HS}_j)}{N_{\text{MH}}}\,, \qquad (5)$$

$$\Pr[\text{Sen}_x \to \text{Rec}_y] = \frac{\sum_{j \in N_{\text{MH}}} I_{\text{Sen}_x \to \text{Rec}_y}(\mathcal{HS}_j)}{N_{\text{MH}}}\,.$$

Our methodology aims to establish whether these probabilities are correct.

Our test consists of running our basic experiment over 2 000 observations. In each of them we select a random input message ($i_x$) and a random output message ($o_y$) as targets and we store the tuple:

$$(\Pr[i_x \to o_y], I_{i_x \to o_y}(\text{trace}))\,.$$

[1]Although in this work we assume $p_{\overline{cp}}$ is known to the attacker, it could be included in the Hidden State and inferred together with the rest of hidden variables.

The first element of the tuple is the inferred probability that $i_x$ corresponds to $o_y$ computed with Eq. 5 from the result of the MH simulation. The second element, $I_{i_x \to o_y}(\text{trace})$, is an indicator variable that takes the value 1 if $o_y$ actually corresponded to $i_x$ when the trace was generated, and 0 otherwise. We note that the test could be also carried on using senders and receivers as targets with the sole difference that the tuples stored would be:

$$(\Pr[\text{Sen}_x \to \text{Rec}_y], I_{\text{Sen}_x \to \text{Rec}_y}(\text{trace}))\,.$$

Once these tuples are collected, we make a histogram with 30 "bins" of equal size using the first element of the tuple as distinguisher for the classification. We denote as $\text{bin}(a, b)$ the "bin" containing $\Pr[i_x \to o_y] : a \leq \Pr[i_x \to o_y] < b$, and $\text{Len}(a, b)$ the number elements in that bin. For each of the bins we compute:

- The value $p_{\text{sampled}}(a, b)$, which corresponds to the mean of the $\Pr[i_x \to o_y]$ belonging to the tuples contained in the bin:
$$p_{\text{sampled}}(a, b) = \frac{\sum_{\Pr[i_x \to o_y] \in \text{bin}(a,b)} \Pr[i_x \to o_y]}{\text{Len}(\text{bin}(a, b))}\,.$$
- $p_{\text{empirical}}(a, b)$, the 95% Bayesian confidence intervals given how many tuples there are on a bin and the amount of these tuples whose second element is $I_{i_x \to o_y}(\text{trace}) = 1$ using the Beta function:
$$\alpha = \sum_{I_{i_x \to o_y} \in \text{bin}(a,b)} I_{i_x \to o_y}(\text{trace}) + 1\,,$$
$$\beta = \text{Len}(\text{bin}(a, b)) - \alpha + 2$$
$$p_{\text{empirical}}(a, b) \sim \text{Beta}(\alpha, \beta)\,.$$

The value $p_{\text{sampled}}(a, b)$ represents the expected probability for an event given the MH simulation output (Eq. 5.) The Bayesian confidence interval $p_{\text{empirical}}(a, b)$ represents the "actual" probability with which the targeted events happened in the observations.

We expect the mean $p_{\text{sampled}}(a, b)$ to fall within the interval $p_{\text{empirical}}(a, b)$, i.e. the estimated probability being close to the probability with which events happen in the generation of the traces. If this is the case we conclude that the implementation of the Metropolis-Hastings sampler is correct. The size of the confidence interval is also meaningful: Small intervals indicate that many samples have been used thus, it accurately represents $p_{\text{empirical}}(a, b)$. On the other hand, if few samples are used to compute the interval (if a bin contains few events), we obtain a poor estimate of $p_{\text{empirical}}(a, b)$ and the results based on it are rather meaningless.

## 5.3 Evaluation Results

We conducted several experiments considering both the basic constraints and the full model (including non-compliant clients) in small and large networks.

Figure 6 shows the result of our evaluation using only basic constraints to generate the trace and model it. The lower graph is a histogram of the number of experiments per bin, $\text{Len}(\text{bin}(a, b))$. The upper graph represents with crosses the mean of the bins $p_{\text{sampled}}(a, b)$, and the Bayesian confidence intervals $p_{\text{empirical}}(a, b)$ with vertical lines. Most crosses fall in the intervals, meaning that our algorithm is providing samples $\mathcal{HS}_j$ according to the correct distribution.

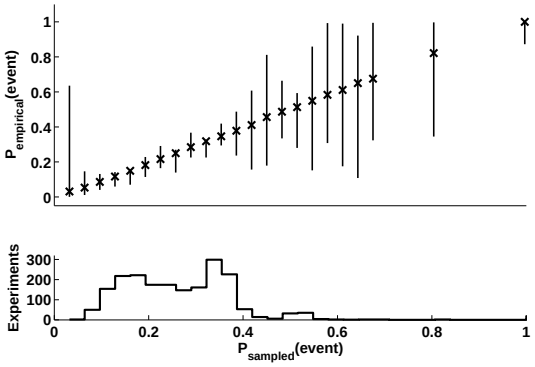



**Figure 6: Results for the evaluation of an observation generated by 50 messages in a network with $N_{\mathbf{mix}} = 3$ and $t = 3$, when all clients behave in a compliant way**

(Only 95% are expected to fall within the intervals.) Most messages fall in bins with $p_{\text{sampled}} \in [0.07, 0.4]$, and their confidence intervals are very small, indicating that we have a high certainty our sampler works correctly in that region.

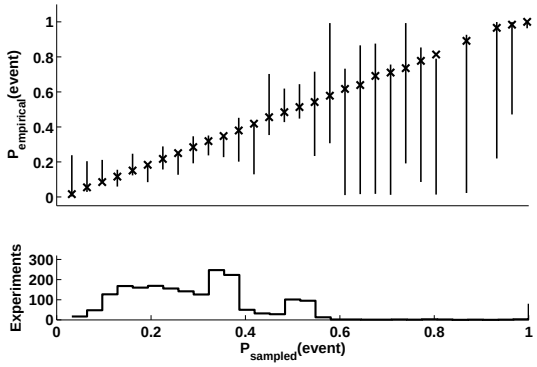
It is noticeable that some paths fall in the $p_{\text{sampled}} = 1$ bin. This denotes total certainty about the correspondence between an input and an output, with no anonymity provided. These are deterministic paths (explained in Sect. 4.1) where the attacker is completely sure that the message $i_x$ corresponds to the potential output message $o_x$ because it is the only message inside a mix.

Experiments were also performed where some of the clients behave in a non-compliant fashion. The result for $N_{\text{msg}} = 10$ messages is shown in Fig. 7(a). We observe more events with $p_{\text{sampled}} = 1$ that represent deterministic paths. This increase is due to long non-complient paths ($L_{x,\overline{cp}} >> L_{\max}$) whose links cannot be swapped to form compliant paths.

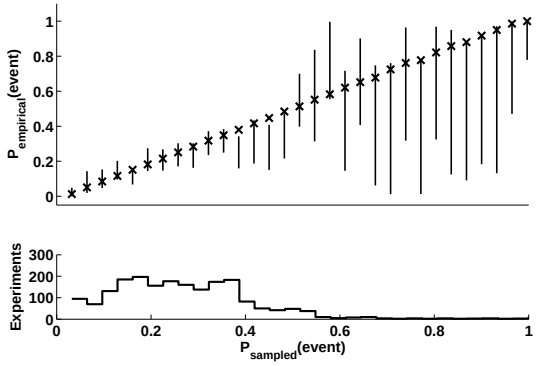
A second difference, with respect to the compliant case, is the appearance of a significant number of events with probability $p_{\text{sampled}} \in [0.7, 1]$. These are paths with no compliant alternative, that now appear as non-compliant paths, with the associated small probability. The probability of these paths is diminished more (generating events with probability $p_{\text{sampled}} \approx 0.7$) or less (generating events with probability $p_{\text{sampled}} \approx 0.95$) depending on how likely the non-compliant path is. These events happen rarely and the number of samples falling in these bins is small, resulting in large confidence intervals.

Figure 7(b) shows our results when considering 50 messages. As one would expect, we can see in the histogram at the bottom that when more messages travel through the network, and the attacker is less certain about their destination. There are also fewer samples in the $p_{\text{sampled}} = 1$ bin, which reflects the increase in the anonymity that the presence of more traffic in the network provides to its users.

Finally, we tested the effectiveness of our sampler for longer observations (100 and 1000 messages in the network.) The results of the experiments are shown in Fig. 8. In these cases, we find that the mix network provides good anonymity for all messages. An attacker cannot link incoming and outgoing messages with a probability higher than $p_{\text{sampled}} = 0.4$ when 100 messages have been observed, and $p_{\text{sampled}} = 0.1$ if more messages are seen.




(a) $N_{\text{msg}} = 10$ messages




(b) $N_{\text{msg}} = 50$ messages

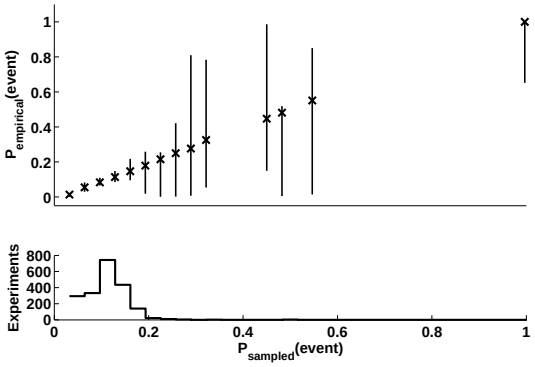
**Figure 7: Results for the evaluation of an observation of a network with $N_{\mathbf{mix}} = 3$ and $t = 3$, when non-compliant clients are present**

In all examples, we obtain the expected result: approximately 95% of the samples fall into the confidence intervals. We conclude that our implementation produces samples from the correct a-posterior probability distribution and implements the optimal Bayesian inference an adversary can perform.

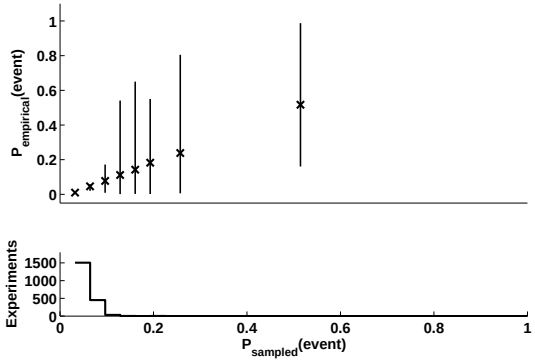
## 5.4 Performance evaluation

Our Metropolis-Hastings sampler is composed by 1443 LOC of Python, including the code associated to the evaluation. Our implementation is not optimised for size, memory usage or running time, and an equivalent implementation in C would outperform it.

The sampler implementation uses a "two-states" strategy for the proposal and acceptance/rejection of candidates $\mathcal{HS}'$. This strategy stores two states $HS_0$ and $HS_1$ that are initialised to the same value (the initial state). In order to propose a candidate we apply a transition $Q$ on $HS_1$, and compute $\alpha$ (considering $\mathcal{HS}_j = \mathcal{HS}_0$ and $\mathcal{HS}' = \mathcal{HS}_1$). If the state is to be accepted, we apply the same transformation to $HS_0$ ($HS_0 = HS_1$). If on the contrary there is a rejection, we undo the modification on $HS_1$ ($HS_1 = HS_0$). Then we restart the process with a new transition $Q$. This strategy apparently doubles the memory requirements, but actually reduces the amount of extra information needed to walk forward and backwards between states, resulting in a smaller total overhead, and significant ease of implementation.




(a) $N_{\text{msg}} = 100$, $N_{\text{mix}} = 10$, $t = 20$




(b) $N_{\text{msg}} = 1000$, $N_{\text{mix}} = 10$, $t = 20$

**Figure 8: Results for the evaluation of big networks**

**Table 2: Metropolis-Hastings RAM requirements**

| $N_{\mathbf{mix}}$ | $t$ | $N_{\mathbf{msg}}$ | **Samples** | **RAM (Mb)** |
|---|---|---|---|---|
| 3 | 3 | 10 | 500 | 16 |
| 3 | 3 | 50 | 500 | 18 |
| 10 | 20 | 100 | 500 | 19 |
| 10 | 20 | 1 000 | 500 | 24 |
| 10 | 20 | 10 000 | 500 | 125 |

The memory requirements of the sampler are well within the range of a commodity computer. Table 2 presents the memory requirements for different sizes of the observation given by the parameters $N_{\text{mix}}$, $t$, and $N_{\text{msg}}$. More memory is needed as observations $\mathcal{O}$ and consequently samples $\mathcal{HS}$ grow. Furthermore, we keep the samples $\mathcal{HS}_i$ in memory, multiplying the overhead for the number of samples collected (double in the case of having 1 000 or more messages with respect to the case when only 50 or 10 messages are considered.)

Finally, we measured the computation time for processing observations of distinct size. For each of the sizes we collected 100 measurements of the analysis time and averaged over them. These timings are shown in Table 3.

Computation time increases as the observations increase for two reasons. First, more iterations $\iota$ are needed to produce independent samples. Second, our the timings include the analysis of all messages in the system, that grow with the observation. Although the time necessary to perform the analysis is already practical, it can be reduced considerably through paralelizing several MH simulations for the same observation to get samples $\mathcal{HS}_j$ faster.

**Table 3: Metropolis-Hastings timings**

| $N_{\mathbf{mix}}$ | $t$ | $N_{\mathbf{msg}}$ | $\iota$ | **Full analysis** (min) | **One sample** (ms) |
|---|---|---|---|---|---|
| 3 | 3 | 10 | 6011 | 4.24 | 509.12 |
| 3 | 3 | 50 | 6011 | 4.80 | 576.42 |
| 10 | 20 | 100 | 7011 | 5.34 | 641.28 |
| 10 | 20 | 1 000 | 7011 | 5.97 | 716.72 |

# 6. MEASURING ANONYMITY

A lot of research has been done regarding the evaluation of anonymity system. Several tools have been proposed to measure the anonymity provided by this systems [3, 12, 32, 18], amongst which the most popular are the metrics based on Shannon entropy [15, 29]. These metrics are computed over the probability distributions associated with random variables representing user's sending profiles, network level profiles (incoming to outgoing messages correspondences), etc. They give a measure of the uncertainty of the attacker about the possible outcome of the random variable under study.

It is important to realise that the methodology presented in this work does not output a probability distribution, but samples that allow us to approximate probabilities of certain events: $\Pr[i_x \to o_y]$, being $i_x$ an incoming message and $o_y$ an outgoing message. However, only events that have been sampled can be estimated, and we cannot assume that not-sampled events have a null probability. After a finite MH simulation there may be events with very small probability which the random walk has not yet visited (or that have been visited but not sampled) but this does not mean that they are impossible to reach. The estimation of probabilities using MH samples introduces an inherent error coming from the normalisation over the sampled events, and not all possible ones. Hence, it cannot be considered a proper probability distribution and it is not possible to measure anonymity by directly applying previously proposed metrics. In this section we explain how to use the MH samples to obtain bounds on the anonymity provided by the system.

Let us consider we want to measure the anonymity provided by the system to a given message $i_x$. We denote the probability distribution of this message corresponding to any of the possible $N$ outgoing message as $\Psi_x = \{\Pr[i_x \to o_y], y = 1, \ldots, N\}$. Following the approach of Serjantov and Danezis [29] we would measure the anonymity for $i_x$ as the Shannon entropy of this probability distribution:

$$H(\Psi_x) = -\sum_y \Pr[i_x \to o_y] \cdot \log \Pr[i_x \to o_y] \,,$$

but as we said we do not have the full probability distribution, and only samples coming from it.

An approach to the estimation of $H(\Psi_x)$ is to model $\Psi_x$ as a multinomial distribution that determines the probability of outputs $o_y$ corresponding to an input $i_x$, and resort again to Bayesian Inference to estimate it from the samples. For this purpose we also define an auxiliary function that counts the number of times a message $i_x$ is assigned to a message $o_y$ in the set of samples, and denote it as $\text{Ct}_{i_x \to o_y}$. We note that the Dirichlet distribution is a conjugate prior

for the multinomial distribution. A sample from this distribution expresses the belief that the probability of the events $i_x \to o_y$ is $\Pr[i_x \to o_y]$ given that we have observed $\mathrm{Ct}_{i_x \to o_y}$ occurrences of each of them. Hence we can use the Dirichlet distribution assuming poor prior knowledge over the actual correspondence (Dirichlet(1,...,1)) to obtain samples from $\Psi_x$ [24]. We compute the entropy $H(\Psi_x)$ of $n$ samples $\Psi_x$ from the posterior distribution:

$$H(\Psi_x) \text{ where } \Psi_x \sim \text{Dirichlet}(\mathrm{Ct}_{i_x \to o_0}+1, \ldots, \mathrm{Ct}_{i_x \to o_N}+1)\,.$$

We note that, for the receivers $o_{\overline{y}}$ that do not appear in the samples, $\mathrm{Ct}_{i_x \to o_{\overline{y}}} = 0$.

We order the samples $H(\Psi_x)$ in decreasing order and take as bounds for the anonymity offered by the system the $\gamma\%$ confidence interval for this distribution, i.e., an interval within the range $[0, 1]$, encompassing $\gamma\%$ of the probability mass of the a-posterior distribution.

# 7. CONCLUSIONS

Each proposed mix system is slightly different from others, and our model has to still be extended to deal with different mixing strategies [27, 30], dummy traffic [8, 14, 27] as well as observations that start while the mix network is running. The model of mix networks is flexible enough to be the basis of such models, although performing efficient inference to estimate the probability of their hidden state might require some craftsmanship.

Beyond mix networks, the 'Holy Grail' of Bayesian traffic analysis would be its application to the setting of low-latency anonymity systems based on onion-routing [31], such as Tor [17]. An adversary in such system is constrained to observe only a fraction of the network, but the observations leak precise cell timing data that can be used to trace streams. Murdoch and Zielinski [26] present a simplified analytical Bayesian analysis in such a setting, under the assumptions that traffic is Poisson distributed. Presenting a general model of an onion routing system, and a practical sampler to perform inference is the next significant step in this line of work.

Our work has demonstrated that we can extract accurate a-posterior distributions about who is talking to whom, from a complex anonymity system, with a vast hidden state-space, and a large observation. For the first time we are able to calculate the distributions necessary to apply any information theoretic or decision theoretic anonymity metrics[13, 11].

Our hope is that the traffic analysis methodology we have employed, that defines a probabilistic model over the full system, and performs Bayesian inference to measure the security of the system, becomes the standard by which candidate anonymity systems are proposed and evaluated. In particular the ability to integrate all information in a traffic analysis, as well as extracting probabilities of error, should be seen as essential for proposing robust attacks.

**Acknowledgements.** The authors would like to thank Steven Murdoch, Tom Chothia, Kostas Chatzikokolakis and Ben Laurie, as well as other participants of the second anonymity meet-up in 2008 for their valuable input on early drafts of this work. C. Troncoso is a research assistant of the Fund for Scientific Research in Flanders (FWO) and this work was partly performed while Carmela Troncoso was an intern at Microsoft Research Cambridge, between September and December 2008. This work was supported in part by the IAP Programme P6/26 BCRYPT of the Belgian State.

# 8. REFERENCES

[1] Dakshi Agrawal and Dogan Kesdogan. Measuring anonymity: The disclosure attack. *IEEE Security and Privacy*, 1(6):27–34, 2003.

[2] David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 4(2), February 1981.

[3] Sebastian Claußand Stefan Schiffner. Structuring anonymity metrics. In *DIM '06: Proceedings of the second ACM workshop on Digital identity management*, pages 55–62, New York, NY, USA, 2006. ACM.

[4] George Danezis. Statistical disclosure attacks: Traffic confirmation in open environments. In Gritzalis, Vimercati, Samarati, and Katsikas, editors, *Proceedings of Security and Privacy in the Age of Uncertainty, (SEC2003)*, pages 421–426, Athens, May 2003. IFIP TC11, Kluwer.

[5] George Danezis and Claudia Diaz. A survey of anonymous communication channels. Technical Report MSR-TR-2008-35, Microsoft Research, January 2008.

[6] George Danezis, Claudia Diaz, and Carmela Troncoso. Two-sided statistical disclosure attack. In Nikita Borisov and Philippe Golle, editors, *Proceedings of the Seventh Workshop on Privacy Enhancing Technologies (PET 2007)*, Ottawa, Canada, June 2007. Springer.

[7] George Danezis, Roger Dingledine, and Nick Mathewson. Mixminion: Design of a Type III Anonymous Remailer Protocol. In *Proceedings of the 2003 IEEE Symposium on Security and Privacy*, pages 2–15, May 2003.

[8] George Danezis and Len Sassaman. Heartbeat traffic to counter (n-1) attacks. In *Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2003)*, Washington, DC, USA, October 2003.

[9] George Danezis and Andrei Serjantov. Statistical disclosure or intersection attacks on anonymity systems. In *Proceedings of 6th Information Hiding Workshop (IH 2004)*, LNCS, Toronto, May 2004.

[10] George Danezis and Paul Syverson. Bridging and fingerprinting: Epistemic attacks on route selection. In Nikita Borisov and Ian Goldberg, editors, *Proceedings of the Eighth International Symposium on Privacy Enhancing Technologies (PETS 2008)*, pages 133–150, Leuven, Belgium, July 2008. Springer.

[11] George Danezis and Carmela Troncoso. The application of bayesian inference to traffic analysis. Technical report, Micorsoft Research, August 2008.

[12] Yuxin Deng, Jun Pang, and Peng Wu. Measuring anonymity with relative entropy. In *Proceedings of the 4th International Workshop on Formal Aspects in Security and Trust*, volume 4691 of *Lecture Notes in Computer Science*, pages 65–79. Springer, 2007.

[13] Claudia Diaz, Steven J. Murdoch, and Carmela Troncoso. Towards practical dependant link padding. *Under submission*, 2009.

[14] Claudia Diaz and Bart Preneel. Reasoning about the anonymity provided by pool mixes that generate dummy traffic. In *Proceedings of 6th Information Hiding Workshop (IH 2004)*, LNCS, Toronto, May 2004.

[15] Claudia Diaz, Stefaan Seys, Joris Claessens, and Bart Preneel. Towards measuring anonymity. In Roger Dingledine and Paul Syverson, editors, *Proceedings of Privacy Enhancing Technologies Workshop (PET 2002)*. Springer-Verlag, LNCS 2482, April 2002.

[16] Claudia Diaz, Carmela Troncoso, and Andrei Serjantov. On the impact of social network profiling on anonymity. In Nikita Borisov and Ian Goldberg, editors, *Proceedings of the Eighth International Symposium on Privacy Enhancing Technologies (PETS 2008)*, pages 44–62, Leuven, Belgium, July 2008. Springer.

[17] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th USENIX Security Symposium*, August 2004.

[18] M. Edman, F. Sivrikaya, and B. Yener. A combinatorial approach to measuring anonymity. *Intelligence and Security Informatics, 2007 IEEE*, pages 356–363, 2007.

[19] Benedikt Gierlichs, Carmela Troncoso, Claudia Diaz, Bart Preneel, and Ingrid Verbauwhede. Revisiting a combinatorial approach toward measuring anonymity. In Marianne Winslett, editor, *Workshop on Privacy in the Electronic Society 2008*, page 5, Alexandria,VA,USA, 2008. ACM.

[20] W. K. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, April 1970.

[21] E. T. Jaynes. *Probability Theory : The Logic of Science*. Cambridge University Press, April 2003.

[22] Dogan Kesdogan, Dakshi Agrawal, and Stefan Penz. Limits of anonymity in open environments. In Fabien Petitcolas, editor, *Proceedings of Information Hiding Workshop (IH 2002)*. Springer-Verlag, LNCS 2578, October 2002.

[23] Dogan Kesdogan and Lexi Pimenidis. The hitting set attack on anonymity protocols. In *Proceedings of 6th Information Hiding Workshop (IH 2004)*, LNCS, Toronto, May 2004.

[24] David J. C. Mackay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003.

[25] Nick Mathewson and Roger Dingledine. Practical traffic analysis: Extending and resisting statistical disclosure. In *Proceedings of Privacy Enhancing Technologies workshop (PET 2004)*, volume 3424 of *LNCS*, pages 17–34, May 2004.

[26] Steven J. Murdoch and Piotr Zielinski. Sampled traffic analysis by internet-exchange-level adversaries. In Nikita Borisov and Philippe Golle, editors, *Privacy Enhancing Technologies*, volume 4776 of *Lecture Notes in Computer Science*, pages 167–183. Springer, 2007.

[27] Luke O'Connor. On blending attacks for mixes with memory. In *Proceedings of Information Hiding Workshop (IH 2005)*, pages 39–52, June 2005.

[28] Andrei Serjantov. *On the Anonymity of Anonymity Systems*. PhD thesis, University of Cambridge, June 2004.

[29] Andrei Serjantov and George Danezis. Towards an information theoretic metric for anonymity. In Roger Dingledine and Paul Syverson, editors, *Proceedings of Privacy Enhancing Technologies Workshop (PET 2002)*. Springer-Verlag, LNCS 2482, April 2002.

[30] Andrei Serjantov and Richard E. Newman. On the anonymity of timed pool mixes. In *Proceedings of the Workshop on Privacy and Anonymity Issues in Networked and Distributed Systems*, pages 427–434, Athens, Greece, May 2003. Kluwer.

[31] Paul Syverson, Gene Tsudik, Michael Reed, and Carl Landwehr. Towards an Analysis of Onion Routing Security. In H. Federrath, editor, *Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, pages 96–114. Springer-Verlag, LNCS 2009, July 2000.

[32] Gergely Tóth and Zoltán Hornák. Measuring anonymity in a non-adaptive, real-time system. In *Proceedings of Privacy Enhancing Technologies workshop (PET 2004)*, volume 3424 of *Springer-Verlag, LNCS*, pages 226–241, 2004.

[33] Carmela Troncoso, Benedikt Gierlichs, Bart Preneel, and Ingrid Verbauwhede. Perfect matching disclosure attacks. In Nikita Borisov and Ian Goldberg, editors, *Proceedings of the Eighth International Symposium on Privacy Enhancing Technologies (PETS 2008)*, volume 5134 of *Lecture Notes in Computer Science*, pages 2–23, Leuven,BE, 2008. Springer-Verlag.

[34] Matthew Wright, Micah Adler, Brian Neil Levine, and Clay Shields. An analysis of the degradation of anonymous protocols. In *Proceedings of the Network and Distributed Security Symposium - NDSS '02*. IEEE, February 2002.

# APPENDIX

## A. A TYPICAL SMALL TRACE

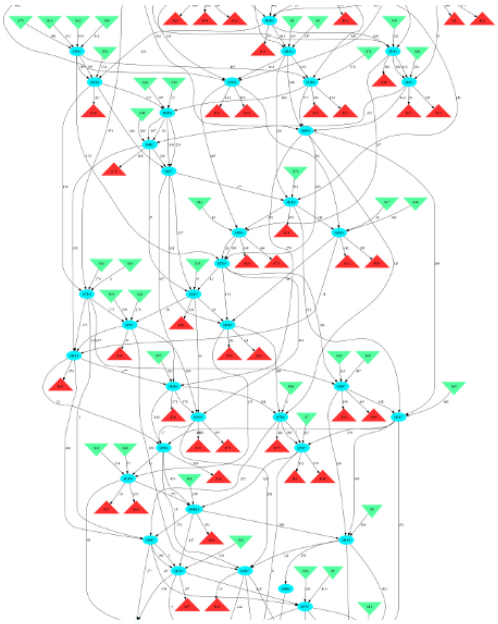


**Figure 9: Fraction of a typical non-toy observation**