# The Application of Bayesian Inference to Traffic analysis

George Danezis Carmela Troncoso

Microsoft Research, Cambridge,
September – December 2008.

## Contents

# 1 Introduction

## 1.1 Anonymity Systems

Anonymous communications allow conversing parties on a network to exchange messages without revealing their network identifiers to each other or to third party observers. This security property is of special importance to ensure privacy, support important protocols such as on-line polls, or enable high-security government or military communications to use commodity network infrastructures.

The most practical proposal for engineering anonymous communications is the *mix*, proposed by David Chaum [9] in 1981 (although early ideas were already in his Masters thesis in 1979). A mix is a network router offering a special security property: it hides the correspondences between its input and output messages, thus providing some degree of anonymity. A large body of research, surveyed in [14], is concerned with extending and refining mix based protocols.

The first engineering challenge to build a mix is to ensure that messages entering and leaving the network are *cryptographically unlinkable*, though a public-key operation inside the mix – most often decryption. Many such cryptographic schemes, also called cryptographic packet formats, have been proposed in the literature [17, 41, 6, 18, 55]. Modern proposals allow the chaining of multiple mixes, without leaking any information about the number of mixes on a path, or the position of each mix. They also allow users to build anonymous addresses that can be used to route mail back to them anonymously by anyone in the network. The cryptographic aspects of mix networks are well understood, and aside from specific efficiency tweaks, mix designers do not have to be concerned by them. This work in its entirety assumes that the cryptographic functions of mixes are secure and do not leak any information.

The difficult design choices mix engineers face relate to achieving *traffic analysis resistance* for mix networks. This means that timing and volume of messages should not betray any information that would allow an adversary to link incoming and outgoing messages of the anonymity infrastructure. Typical mix architectures achieve this by allowing senders to relay their messages over a sequence of mixes, called a *path*. Security has to be achieved under severe constraints: messages have to be routed in a timely fashion [36], the number of connections maintained by each mix is restricted [12], the adversary might be able in inject messages in the network [53], all clients might not know all mix routers [22], rogue mixes might be dropping messages [5], or the architecture may have to be fully peer-to-peer [51, 43]. Complexity of protocols increases to fulfil all these goals, and information may leak as a result about who is talking to whom, through the careful study of traffic patterns and compromised or rogue nodes.

This work is concerned with making use of the traffic patterns of messages as they transit through a mix-based anonymity to infer who is talking with whom.

## 1.2 Traffic analysis

Traffic analysis is a family of techniques used to infer information merely from the meta-data of communications. It can be applied to unprotected or even encrypted [48] communications, like email or web-traffic, to infer users' social networks and interests. In the context of anonymous communications it refers to tracing who is sending messages to whom, therefore compromising their security.

The most naive "brute force" approach to the traffic analysis of mix systems is succinctly presented by J.F. Raymond [49], alongside many basic attacks that later became the focus of intense research:

> 1) The attacker first follows a message from a sender to a first mix node.
> 2) The attacker then follows every ($t$) message that the first node releases. The adversary needs to follow messages going to anywhere between $t$ and 1 different nodes. If all messages are sent to either the same mix node or recipients, the attacker only needs to monitor one node. On the other and, if all h messages are sent to different nodes, the attacker needs to observe $t$ different mix odes.
> 3) The process continues like this until messages reach the $d^{\text{th}}$ level nodes. The attacker then need only "follow" messages leaving the mix network (i.e. going to recipients).

In fact the adversary is trying all possible paths a message could have taken starting at a particular sender – a process that grows exponentially in the branching factor $t$ and the depth of the search $d$. This simple minded attack ignores some important information, that could lead to better and more efficient tracing of messages.

First, paths are chosen by users or the system using some constraints besides length. It is therefore possible to prune paths that are simply not possible. Other path properties have been looked at like the positions of the nodes on the path [3], partial knowledge of the clients [23] (fingerprinting & bridging), special selection of initial nodes [47] (guards), and social relationships between senders and receivers [27]. Some constraint are soft, making paths less likely than others rather than impossible. So far traffic analysis attacks have ignored subtle differences in the likelihood of paths and most research has focused on a binary security question: "Is there a possible unique path or not?" The techniques we present allow all these restrictions to be integrated in the same framework, and used together, and we are able to take into account probabilistic differences and soft constraints.

Second, the brute force approach proposed concentrates on the path of one user, without taking into consideration what others do. This is a mistake: another user may for some reason be attributed some paths, reducing the search complexity to perform the brute force attack. It might also be the case that there is no other user in the system that is able to be ascribed a particular path, making it imperative that it belongs to a target users. Such epistemic reasoning has been explored in specific attacks in the past [23], but our work integrates it

throughout.

Beyond tracing single messages techniques have been developed to uncover persistent and repeated patterns of communication. Such attacks were first named "intersection attacks" [49] since they were based on the idea that a target user systematically communicates with a single friend. By intersecting the anonymity sets of the send messages the friend would be uncovered. Kesdogan *et al.* [34, 2, 37] introduced a family of disclosure and hitting set attacks that generalises this idea to users with multiple friends. Their result, in the long run, is the set of friends being uncovered and many communications being traced. Statistical variants of these attacks were also developed, known as statistical disclosure attacks, and applied to pool mixes [20] and traffic containing replies [15]. The state of the art in statistical disclosure is the Perfect Matching attack introduced by Troncoso *et al.* [61], that introduces models that are the starting point for our Bayesian disclosure attack.

## 1.3 Measuring anonymity

The anonymity provided by a system is related to the uncertainty an adversary has about who is actually communicating with whom. The two established information theoretic metrics [52, 26] measure this uncertainty directly using the concept of entropy. The probability distribution over all possible receivers of a message is calculated, and the entropy of this distribution (normalised for [26]) is the measure of anonymity for that *particular message* and that *particular adversary*. These metrics are widely accepted but seldom used for theoretical as well as practical reasons.

The theoretical difficulties associated with the information theoretic metrics is that they describe the anonymity of single messages, but not systems as a whole. This has sparked a debate on whether the minimum, average, maximum, or typical anonymity of messages should be used to describe a system as a whole. The metrics are also tied in with the specific knowledge of the adversary. It is very difficult for a system designer to calculate them correctly since they only have the most generic threat model, and cannot know exactly what information the adversary has. These theoretical discussion have sparked a lot of research [58, 45, 59, 56, 29, 8, 7] into alternative ways of measuring anonymity, without any specific proposal establishing itself as a preferable alternative yet.

The most serious impediment to applying information theoretic metrics for anonymity, is the difficulty of calculating the probability distributions upon which they rely. Both metrics assume that a probabilistic mapping between a sent message and all potential receivers is at hand – in fact this can only the result of a difficult traffic analysis step. Given these difficulties some authors use heuristics instead of probabilities to extract a mapping – this invariably leads to results that are difficult to interpret [27]. Other works give up on the information theoretic metrics and use the rank (or guessing probability) of the actual sender [16], or the probability of full compromise [22], as measures of anonymity.

The current work directly addresses the hard problem of calculating the

probability distributions over receivers of messages (or senders of messages). We show that a combination of probabilistic modeling, Bayesian inference, and sampling techniques can be used to extract those distributions from observations of rather complex systems. These in turn can be used to compute the information theoretic anonymity of a system.

## 1.4 Bayesian Inference and Monte Carlo Methods

Bayesian inference is a branch of statistics with applications to machine learning and estimation [39]. Its key methodology consists of constructing a full probabilistic model of 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 $\Pr[\mathcal{HS}, \mathcal{O}|\mathcal{C}]$ given a particular model $\mathcal{C}$. By applying Bayes rule we can find the distribution of the hidden state given the observations as:

$$\Pr[\mathcal{HS}, \mathcal{O}|\mathcal{C}] = \Pr[\mathcal{HS}|\mathcal{O}, \mathcal{C}] \cdot \Pr[\mathcal{O}|\mathcal{C}] \Rightarrow \Pr[\mathcal{HS}|\mathcal{O}, \mathcal{C}] = \frac{\Pr[\mathcal{HS}, \mathcal{O}|\mathcal{C}]}{\Pr[\mathcal{O}|\mathcal{C}]} \Rightarrow$$

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

The joint probability $\Pr[\mathcal{HS}, \mathcal{O}|\mathcal{C}]$ is 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 quantity $\mathcal{Z}$ is simply a normalising factor.

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

- 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 steps are, in theory, easily derived from this forward model. In practice computational limitations require careful crafting of the models and the inference techniques to be able to handle large systems.

- The Bayesian approach allows to infer as many characteristics of the system as needed by introducing them in the probabilistic model. This permits to infer several hidden variables jointly as we show for users' sending profiles and their recipient choices for each message.

- A Bayesian treatment results in probability distributions over all possible hidden states, not only the most probable one as many current traffic analysis methods do. The marginal distributions over different aspects of the hidden state can be used to measure the certainly of the attacker, and provide good estimates of her probability of error.

The last point is the most important one: the probability distribution over hidden states given an observation, $\Pr[\mathcal{HS}|\mathcal{O},\mathcal{C}]$, contains a lot of information about all possible states. When traffic analysis is used operationally the probability of error of particular aspects of the hidden state can be calculated to inform decision making. It is very different to assert that, in both cases, the most likely correspondent of Alice is Bob, with certainty 99% versus with certainty 5%. Extracting probability distributions over the hidden state allows us to compute such error estimates directly, without the need for an ad-hoc analysis of false positives and false negatives. Furthermore, the analyst can use the inferred probability distribution to calculate directly anonymity metrics [26, 52].

Despite their power Bayesian techniques come at a considerable computational cost. It is often not possible to compute or characterise directly the distribution $\Pr[\mathcal{HS}|\mathcal{O},\mathcal{C}]$ due to its complexities. In those cases sampling based methods are available to extract some of its characteristics. The key idea is that a set of samples $\mathcal{HS}_0, \ldots, \mathcal{HS}_\iota \sim \Pr[\mathcal{HS}|\mathcal{O},\mathcal{C}]$ are drawn from the a-posterior distribution, and used to estimate particular marginal probability distributions of interest. For this purpose, Markov Chain Monte Carlo methods have been proposed. These are stochastic techniques that perform 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 $\Pr[\mathcal{HS}|\mathcal{O},\mathcal{C}]$. Once the Markov Chain has been built, samples of the hidden states of the system can be obtained by taking the current state of the simulation after a certain number of iterations.

### 1.4.1 Metropolis-Hastings Sampler

The Metropolis-Hastings (MH) algorithm 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. In fact the MH algorithm is closer conceptually to optimization and search algorithms, but outputs samples from a particular distribution instead of a maximal element or a key element for the other two schemes respectively.

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 $\mathcal{HS}' \sim Q(\mathcal{HS}'|\mathcal{HS}_j)$. 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 candidates 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. More samples can be extracted by repeating this process.

This process 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 practice it has to be very easy to compute and sample, as well as being fast mixing, to reduce the number of iterations needed to produce independent samples.

Note that 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, that is simplified when calculating $\alpha$. This is very important as the normalising factor $\mathcal{Z}$ is in practice very difficult to calculate. As all probabilities become very small it is often easier to calculate $\log\alpha$ and test whether $\log\alpha \geq 0$.

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. The number of iterations has to be high enough to ensure the output samples are statistically independent. Calculating it exactly is difficult so we use conservative estimates to ensure we get good samples. The number of MH samples on the other hand depends on the marginal distributions that need to be estimated, and can be increased by running the sampler longer.

An important feature of the MH method is that it can be run in parallel on multiple processor cores or a distributed cluster: all processes output samples that can be aggregated and analysed centrally. Our experiments made use of this property, and we were often running four independent processes on a shared 8 core processor.

### 1.4.2 Gibbs Sampler

The Gibbs sampler [32] is a Markov Chain Monte Carlo method to sample from joint distributions that have easy to sample marginal distributions. These joint distributions are often the a-posterior distribution resulting from the application of Bayes theorem, and thus Gibbs sampling has been extensively used to solve Bayesian inference problems. The operation of the Gibbs sampler is often referred to as *simulation*, but we must stress again that it is unrelated to simulating the operation of the system under attack.

For illustration purposes we assume an a-posterior distribution $\Pr[\mathcal{HS}|\mathcal{O},\mathcal{C}]$ can be written as a joint probability distribution $\Pr[X,Y|\mathcal{O},\mathcal{C}]$ that is difficult to sample directly. If, on the other hand, there is an efficient way of sampling from the marginal distributions $\Pr[X|Y,\mathcal{O},\mathcal{C}]$ and $\Pr[Y|X,\mathcal{O},\mathcal{C}]$, then Gibbs sampling is an iterative technique to draw samples from the joint distribution $\Pr[X,Y|\mathcal{O},\mathcal{C}]$. The algorithm starts at an arbitrary state $(x_0,y_0)$. Then it iteratively updates each of the components through sampling from their respective distributions, i.e. $x_i \sim \Pr[X|Y=y_{i-1},\mathcal{O},\mathcal{C}]$, and $y_i \sim \Pr[Y|X=x_i,\mathcal{O},\mathcal{C}]$.

After a sufficient number of iterations, the sample $(x_i, y_i)$ is distributed according to the target distribution, and the procedure can be repeated to draw more samples. We note that in this process the computation of the normalising factor $\mathcal{Z}$ is not needed.

The other parameters of the Gibbs algorithm, namely the number of iterations necessary per sample, as well as the number of samples are also of some importance. The number of iterations has to be high enough to ensure the output samples are statistically independent. Calculating it exactly is difficult so we use conservative estimates to ensure we get good samples. As for the MH algorithm, the number of samples to be extracted, depends on the necessary accuracy when estimating the marginal distributions, which can be increased by running the sampler longer.

### 1.4.3 Bayesian Confidence Intervals & Probability Estimation

In this work we use Bayesian methods to perform traffic analysis, i.e. to extract samples of the hidden state according to an a-posterior distribution. Furthermore, we use Bayesian confidence intervals to estimate the errors of our estimates for probabilities and other quantities of interest, often extracted from samples $\mathcal{HS}_0, \ldots, \mathcal{HS}_l \sim \Pr[\mathcal{HS}|\mathcal{O}, \mathcal{C}]$ produced by a Metropolis-Hasting or Gibbs algorithm.

Many quantities we estimate represent probabilities. A typical example is the probability a particular sender Alice has sent a message to a particular user Bob, denoted $\Pr[A \rightarrow B] = p_{A \rightarrow B}$. We denote this event as "$A \rightarrow B$" and define two indicator variables $I_{A \rightarrow B}(\cdot)$ and $I_{A \not\rightarrow B}(\cdot)$ taking value one when a sample state contains or not that event respectively. Otherwise they take the value zero.

The task of estimating $p_{A \rightarrow B}$ from the samples $\mathcal{HS}_0, \ldots, \mathcal{HS}_l$ is equivalent to estimating the bias of a weighted coin given a sequence of independent tosses. The forward probability $\Pr[I_{A \rightarrow B}(\mathcal{HS}_0), \ldots, I_{A \rightarrow B}(\mathcal{HS}_l)|p_{A \rightarrow B}]$ is a Bernoulli distribution. The inverse probability $\Pr[p_{A \rightarrow B}|I_{A \rightarrow B}(\mathcal{HS}_0), \ldots, I_{A \rightarrow B}(\mathcal{HS}_l)]$, given a prior distribution Beta with parameters $\text{Beta}(1, 1)$ is:

$$p_{A \rightarrow B} \sim \text{Beta}(\sum_{\forall \mathcal{HS}_j} I_{A \rightarrow B}(\mathcal{HS}_j) + 1, \sum_{\forall \mathcal{HS}_j} I_{A \not\rightarrow B}(\mathcal{HS}_j) + 1)$$

Instead of using a point estimate for the mean of this distribution we often use a confidence interval within the range $[0, 1]$, encompassing 95% or 99% of the probability mass of the a-posterior distribution. When conditioning on infrequent events, we observe that the number of samples used may be quite small, and the confidence intervals are large, warning the user not to put too much faith in the results. Most often we can increase the number of sampled states to achieve a very high confidence in the results of the analysis.

This binomial model, and the corresponding inference on its probability parameter, is used through our work, both to estimate probabilities of interest to the traffic analyst as well as for evaluating the correctness of our sampler.

# 2 Probabilistic Models of Anonymity Systems

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.

## 2.1 Simple Mix Network Model

We consider an anonymity system formed by $N_{mix}$ threshold mixes [9]. 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 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 $\mathcal{O}$. To achieve this it is sufficient to guess 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:

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

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

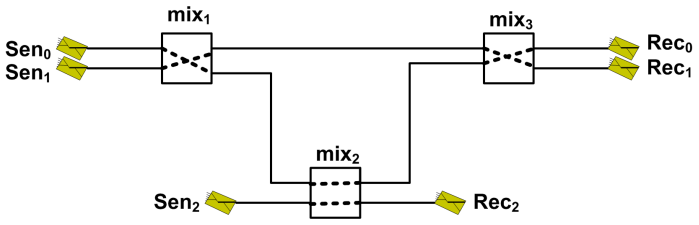



Figure 1: Observation of the network and Hidden State

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:

$$\Pr[\mathcal{HS}|\mathcal{O}, \mathcal{C}] = \frac{\Pr[\mathcal{P}|\mathcal{C}]}{\mathcal{Z}}, \tag{1}$$

where $\mathcal{Z}$ is a normalising constant.

**Proof:** Given that the probability of a path is restricted by the constraints $\mathcal{C}$ and that $\Pr[\mathcal{O}, \mathcal{C}]$ is a constant we obtain the following equation:

$$\Pr[\mathcal{P}|\mathcal{C}] = \Pr[\mathcal{O}, \mathcal{HS}|\mathcal{C}] = \Pr[\mathcal{HS}|\mathcal{O}, \mathcal{C}] \cdot \Pr[\mathcal{O}, \mathcal{C}]$$

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

One can sample the distribution of hidden states $\Pr[\mathcal{HS}|\mathcal{O}, \mathcal{C}]$ by sampling the distribution of paths $\Pr[\mathcal{P}|\mathcal{C}]$ using Bayesian inference techniques. In this particular case we use the Metropolis-Hastings method explained in Sect. 1.4.1. In the next sections we present a probability model that can characterise the distribution of paths under different system-based and user-based constraints.

## 2.2 Basic Constraints

First, we first 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

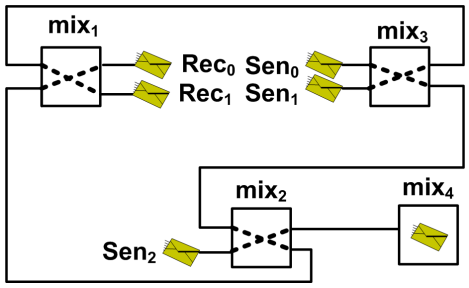



Figure 2: Example where some mixes do not flush

distribution of path lengths, and any 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:

$$\Pr[P_x|\mathcal{C}] = \Pr[L_x = l|\mathcal{C}] \cdot \Pr[M_x|L_x = l, \mathcal{C}] \cdot I_{\text{set}}(P_x) , \tag{2}$$

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 $L_2 = 3$.

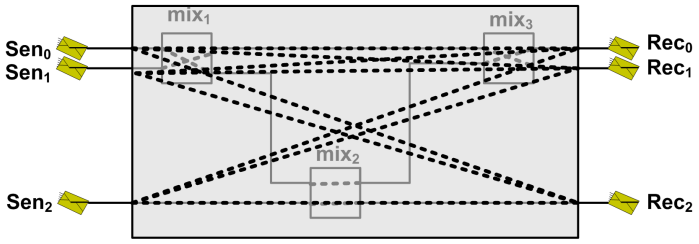



Figure 3: Black box abstraction of the system

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

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

### 2.2.1 Including Social Information

So far we only considered the routing decisions made by the users, but we must not forget that assignments of senders to recipients affect also the probability of hidden states. One can abstract the system as a black box to which users send (and from which users receive) 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.

The number of permutations $\pi$ of $N_{\text{msg}}$ messages is $N_{\text{msg}}!$. Without any a-priori information, the probability of the real permutation $\pi_i$ being any of them is: $\Pr[\pi_i] = 1/N_{\text{msg}}!$. This information can be integrated in the computation of the probability of a set of paths:

$$\Pr[\mathcal{P}|\mathcal{C}] = \prod_{x=1}^{N_{\text{msg}}} \Pr[P_x|\mathcal{C}] \cdot \frac{1}{N_{\text{msg}}!}\,. \tag{4}$$

## 2.3 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.

### 2.3.1 Bridging

Bridging attacks were proposed by Danezis and Syverson in [23]. 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 one if all mixes in a given path $P_x$ are known to the initiator of the path, and is set to zero otherwise. We can easily integrate bridging in Eq. 2:

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

This probability can in turn be used in Eq. 4 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 [23]. 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 [62] in Tor [28] can be modelled in this manner.

### 2.3.2 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 [28] standard client will choose paths of length three as well as distinct onion routers. Furthermore the first router will be a "guard" [62] 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., $L_{\overline{cp}} \in [L_{\min_{\overline{cp}}}, L_{\max_{\overline{cp}}}]$ 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:

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

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. 2.

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. 2.2 and derive:

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

### 2.3.3 Integrating Social Network Information

A number of attacks, starting with Kesdogan *et al* in [35, 1], and further studied in [13, 21, 38, 40, 16, 60], 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. We are initially concerned with incorporating this information into our basic model, and defer until Sect. 4 the discussion of how to extract those profiles.

Let us assume that each sender $\text{Sen}_x$ can be associated with a sending profile, i.e., a probability distribution where each element $\Pr[\text{Sen}_x \rightarrow \text{Rec}_y]$ expresses the probability of sender $\text{Sen}_x$ choosing $\text{Rec}_y$ as the recipient of a message. We propose two conceptual approaches to use this information, and we note that they are mathematically equivalent.

The first option is to incorporate the profile information when computing $\Pr[\pi_i]$. In 2.2.1 we considered that, without any further information, all possible permutations are equally likely. Users profiles, however, increase and decrease the likelihood of some permutations. If we express the black-box model (see

Fig. 3) as a bipartite graph [30, 60], in which an edge going from $\mathrm{Sen}_x$ to $\mathrm{Rec}_y$ ($\mathrm{Sen}_x \to \mathrm{Rec}_y$) has a weight $\Pr[\mathrm{Sen}_x \to \mathrm{Rec}_y]$, a permutation of the input messages into output messages is equivalent to a perfect matching in the underlying graph. Assuming that users make independent decisions on their receivers, the joint probability of the permutation is the product of the individual edge probabilities in this matching [60]:

$$\Pr[\pi_i] = \prod_{(\mathrm{Sen}_x \to \mathrm{Rec}_y) \in \pi_i} \Pr[\mathrm{Sen}_x \to \mathrm{Rec}_y] \,.$$

This probability can be in turn used in Eq. 4 to obtain the probability of a hidden state.

A second approach includes the information derived from the profiles on the path probability calculation. In this case, Eq. 2 becomes:

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

$\mathrm{Sen}_x$ being the originator of the path $P_x$ and $\mathrm{Rec}_y$ the recipient of her message. This probability is in turn used in Eq. 3 to calculate the probability of a hidden state. Note that Eq. 4 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.

#### 2.3.4 Other Constraints

The model proposed in this section is very rich and encompasses aspects of mix based communications never before unified under a common framework. Its clear structure can further be used to incorporate aspects of anonymous communications that have not been taken into account:

- Guard nodes [47], as used by the Tor [28] path selection algorithm are a small set of nodes per user that are used to build the first hop of a tunnel. Since they are part of the path selection algorithm they can be incorporated into the model as a special case of bridging.

- Other mixing strategies can be incorporated into the model besides the traditional threshold mix considered so far. The technique of Serjantov and Newman [54] can readily be adapted to model pool mixes in the current model: each round of the pool mix is represented as a separate threshold mix, where some of the messages (the pool) simply transit from one round to the next. The only modification to the current model is for this transition, from one round to another round of the same mix, not to increase the length of the path.

  More complex mix strategies require more state to be held per mix, and some of them require inference of this hidden state. In some sense this would set in the Bayesian framework inference attacks already described in [46].

- Dummy messages generated by mixes to foil traffic analysis can be incorporated in the model, by simply guessing which messages are dummies, and describing the probability of their paths. This can be useful for foiling the protection afforded by rgb-mixes [19] and active mixing strategies [25, 46].
- The adversary may control some mixes, or inject messages through known paths. Such attacks can be easily modelled by simply moving all the known information from the hidden state to the observations. In such cases it is not necessary to take into account what happened in the likelihood estimation, unless its probabilities change based on the hidden state.
- Finally we have assumed that the start of all paths is known, even though the observation may be truncated before the end of the path is observed. Other models of partial network observation can also be envisaged: the adversary might just be able to observe a window of time, or only some links in the network. Models that extend the concepts of "unknown" sources or sinks of traffic can be build for these circumstances.

# 3 Bayesian Inference & Traffic Analysis

Given an observation $\mathcal{O}$ of some messages' flow in an anonymity network and some knowledge about its functioning and its users' behaviour $\mathcal{C}$, the traffic analysis problem consists in uncovering the relation between senders and receivers, or equivalently in finding the links between incoming and outgoing messages in the system. This can be seen as obtaining the a-posteriori 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. 2.1 that we can sample states $\mathcal{HS} \sim \Pr[\mathcal{P}|\mathcal{C}]$ (see Eq. 1.) These samples can in turn be used to infer the probability distributions that describe events of interest in the system. For instance, given the samples it is easy to estimate the marginal probability distributions $\Pr[i_x \to o_y|\mathcal{O},\mathcal{C}]$ 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. Furthermore, we can estimate the sampling error and provide a confidence interval on the probability obtained.

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 $I_{\text{Sen}_x \to \text{Rec}_x}(\mathcal{HS}_j)$.

We use the Metropolis-Hastings (MH) algorithm, as presented in Sect. 1.4.1, to extract samples from $\Pr[\mathcal{P}|\mathcal{C}]$ following the probability model described in Sect. 2.1.

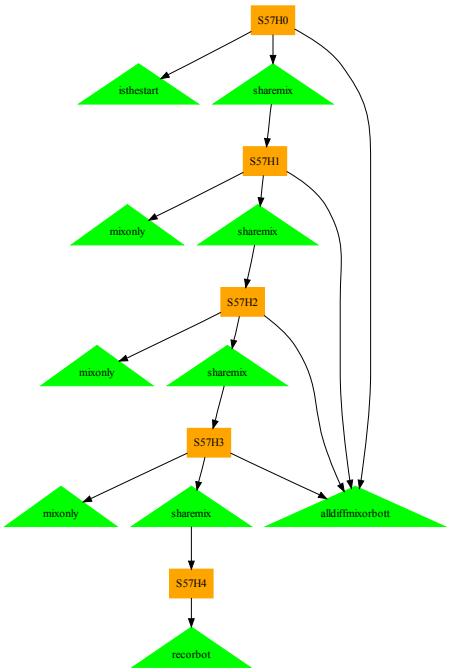



Figure 4: Constraint programming model for a mix network with basic constraint.

Finally, we explain how our results can be combined with the approach presented in [26, 52] to measure the anonymity provided by an anonymity system. The authors suggest to use the Shannon's entropy of these probability distributions, $\Pr[i_x \rightarrow o_y|\mathcal{O},\mathcal{C}]$ and $\Pr[\text{Sen}_x \rightarrow \text{Rec}_y|\mathcal{O},\mathcal{C}]$, to evaluate the anonymity provided by a system. However, contrary to the assumption made in [26, 52] where an adversary has full knowledge about the probability distributions of events in the system, the Metropolis-Hastings simulation only provides an estimation of this distributions because the least likely events are most probably not sampled. Thus, we can only provide bounds on the anonymity that can be expected when using the system. We explain how to compute these bounds in Sect 3.3.

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 $\Pr[i_x \rightarrow o_y]$ when we refer to $\Pr[i_x \rightarrow o_y|\mathcal{O},\mathcal{C}]$) unless stated differently.

## 3.1 Constraint satisfaction problems

A key problem when using a Metropolis-Hasting Sampler is that we need to start the sampler at an arbitrary valid state. Finding such a state given a trace of a heavily constrained mix network is not a trivial task. For example if all path lengths are fixed to be of length three, it might not be trivial to find a hidden state that assigns paths of exactly that length.

One approach to find such a valid initial state is to use a constraint solver. A thorough review of the constraint satisfaction literature is provided in [4]. Constraint solvers take a set of constraints on variables and produce an assignment of values that satisfies those constraint.

Figure 4 illustrates the structure of the constraints on variables for creating valid paths of certain lengths. Each of the variables is a link we observed. Paths are formed by assigning links in a specific structure that guarantees a certain maximum and minimum length, as well as continuity. The constraint for each path is the first link to start at a sender and the last links to end at a receiver depending on the length sought. Further, the intermediate links must share a mix node (to be a continuous path), and we can constrain them to only start and end at a mix – to avoid shorter paths. Finally, the global constraint that all assignments of links to variables is necessary to ensure that all links are only used once. Such all-different constraints can be implemented using Regin's algorithm [50].

While the model is correct, and our toy CSP solver did find assignments for small traces, we found this approach unnecessary for the full model of the mix network. Inconsistent paths in the full model, ensure that even very short or very long paths have some support, and every invalid path, can be made valid by being tagged as inconsistent. The fact that all our constraints are 'soft' makes the use of a constraint solver unnecessary. When constraints are hard, the proposed model in Figure 4 could be vital.

## 3.2 Metropolis-Hastings Sampler

Let us consider an anonymity network where users behave as described in Sect. 2.1. An instance of such a network where 10 messages are sent through 3 mixes of threshold $t = 4$ can be seen in Fig. 5. In this figure, senders are represented as triangles and labeled "SX", X being their identity. Likewise for receivers, represented as triangles labelled "RX". 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 "MXRY", where X expresses the identity of the mix and Y the round of flushing. (A non-toy example of a trace can also be seen in Figure 14 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

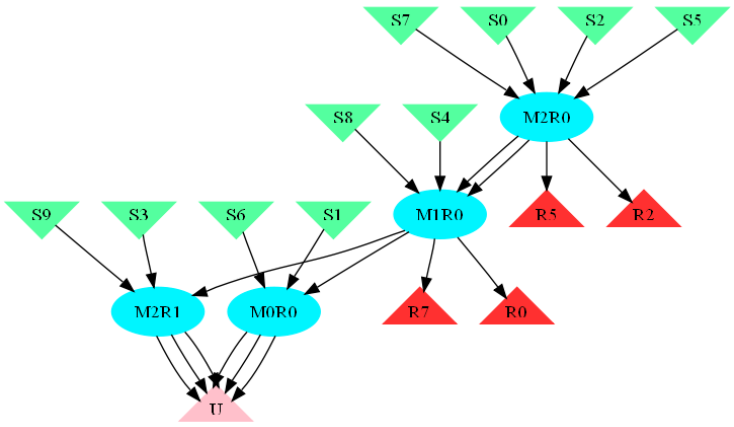



Figure 5: Observation of a network where 10 messages are sent to a network composed by 3 mixes of threshold $t = 4$

mixes "virtual mixes" and denote the set they form as vmixes (in the example vmixes $= \{\text{M0R0}, \text{M1R0}, \text{M2R0}, \text{M2R1}\}$)

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. 2.1 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. 2.2) 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 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. 6.)

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. 1.4.1.)


Figure 6: $Q_{\text{swap}}$ transition operation on the second and third links of a mix

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

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

If we move beyond the basic model to take 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 labeled as compliant or non-compliant ($\text{Sen}_{x,cp}$ or ($\text{Sen}_{x,\overline{cp}}$, respectively) resulting in different probability for the path $P_x$ it starts, and hence leading to different hidden state probabilities $\Pr[\mathcal{HS}]$. We augment the hidden state to be the internal connections in the virtual mixes, together with the set of path labels indicating whether the users initiating the path are compliant with the system or not. In the augmented model the random walk must not only consider swaps of connections, but also changes in the path's initiator 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 must consider changing 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:

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

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

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

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

When the adversary constructs an observation of a network, there may be input messages for which there is certainty about which is the output message they correspond to. These are messages belonging to a one-hop path resulting

from a sender being the only one in choosing a mix as first hop in her path that never flushes in the observation period. We call these "deterministic paths", as the attacker can, without performing any analysis, assign input to output message with probability $\Pr[i_x \to o_y] = 1$ ($o_y$ is the only message in the mix where $i_x$ has arrived.) These paths will never participate in a swap operation as there is only one message in the mix thus its assignment cannot be swapped. 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 ($\Pr[Q_{\text{det}}] = 0$.)

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

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

### 3.2.1 Evaluation

The aim of our evaluation is to ensure that the inferences drawn from the metropolis-hasting samples are "correct". The key to correctness is that the a-posterior distributions returned represent indeed the probabilities of paths, and correspondences between senders and receivers, in the system. In order to evaluate the Metropolis-Hastings sampler described in the previous section, we implement it in the Python language. We consider small (3 mixes) and large (5 to 10 mixes) networks for the evaluation. 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 $p_{\overline{cp}} = 0.1$.

**Basic experiment.** For a given observation, we collect $N_{\text{MH}}$ samples of the distribution $\Pr[\mathcal{HS}]$ using the Metropolis-Hasting algorithm with the transitions $Q$ described in the previous section. 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 [33] 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:

$$\Pr[i_x \to o_y] = \frac{\sum_{j \in N_{\text{MH}}} I_{i_x \to o_y}(\mathcal{HS}_j)}{N_{\text{MH}}}\,, \tag{5}$$
$$\Pr[\text{Sen}_x \to \text{Rec}_y] = \frac{\sum_{j \in N_{\text{MH}}} I_{\text{Sen}_x \to \text{Rec}_y}(\mathcal{HS}_j)}{N_{\text{MH}}}\,.$$

**Metropolis Hastings parameters choice.** The sampler parameters are key for the correctness of the samples, i.e., to ensure that the $\mathcal{HS}$ returned by the sampler come from the desired distribution $\Pr[\mathcal{HS}]$.

The number of iterations $\iota$ must guarantee the independence of the samples. There is not any straightforward procedure to obtain the optimal value for the parameter $\iota$. We consider $\iota$ to be appropriate when it is large enough for the second order statistics of the marginal probability distributions $\Pr[i_x \to o_y]$ (respectively $\Pr[\text{Sen}_x \to \text{Rec}_y]$) to be 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 which was the corresponding output of $i_x$ at sample $j-1$. Formally, the property we are looking for is:

$$\Pr[i_x \to o_y \text{ in } \mathcal{HS}_j | i_x \to o_h \text{ in } \mathcal{HS}_{j-1}] = \Pr[i_x \to o_y \text{ in } \mathcal{HS}_j]\,, \tag{6}$$

for some $h$.

There is no analytic procedure to obtain the optimal value for the parameter $\iota$, so we experimentally test subsequent samples for independence to select the value for $\iota$. We conducted the following independence test on networks with $N_{\text{msg}} = 10$, $N_{\text{mix}} = 3$ and $t = 2$. We collect $N_{\text{MH}} = 2500$ Metropolis-Hastings samples separated by a small number of iterations ($\iota = 509$). In each of the networks we pick at random an input message $i_x$, and obtain its second statistic vector: $V = \{(o_h, o_y).\, \forall l \geq 1 \wedge I_{i_x \to o_h}(\mathcal{HS}_{l-1}) \wedge I_{i_x \to o_y}(\mathcal{HS}_l)\}$. Then, we choose a receiver $o_y$ and compute the first and second order probability that $i_x$ corresponds to $o_y$:

$$p_1 = \Pr[i_x \to o_y] = \frac{\sum_{j \in N_{\text{MH}}} I_{i_x \to o_y}(\mathcal{HS}_j)}{N_{\text{MH}}}\,.$$
$$p_2 = \Pr[i_x \to o_y \text{ in } \mathcal{HS}_j | i_x \to o_h \text{ in } \mathcal{HS}_{j-1}] = \frac{\sum_{l=1,\ldots,(N_{\text{MH}}-1)} I_{(*,o_y)}(V_l)}{N_{\text{MH}} - 1}\,.$$

In order to determine if these probabilities are equal, we must account for the sampling error. Direct comparison is not an option since they will always differ in some larger or smaller amount. Instead, we consider the posterior distribution of $p_1$ and $p_2$ given the data obtained in the experiment and a uniform prior distribution $\text{Beta}(1,1)$. Respectively, these distributions are:

$$\text{Beta}_1\Big(\sum_{j \in N_{\text{MH}}} I_{i_x \to o_y}(\mathcal{HS}_j) + 1, N_{\text{MH}} - \sum_{j \in N_{\text{MH}}} I_{i_x \to o_y}(\mathcal{HS}_j) + 1\Big)\,,$$

and

$$\mathrm{Beta}_2(\sum_{l\in(N_{\mathrm{MH}}-1)} I_{(*,o_y)}(V_l)+1, N_{\mathrm{MH}}-1-\sum_{l\in(N_{\mathrm{MH}}-1)} I_{(*,o_y)}(V_l)+1)\,,$$

(see [31] for further details.) We then test if $\mathrm{Beta}_1$ and $\mathrm{Beta}_2$ are the same distribution (or sufficiently near) by checking that the difference between pairs of samples from these distributions is zero in average. For this purpose we collect 10 000 samples of the difference between them ($d_m = r_m - q_m\,, r_m \in \mathrm{Beta}_1\,, q_m \in \mathrm{Beta}_2\,, m = 1, \ldots, 10\,000$), obtain the 50% Bayesian confidence intervals (i.e., the 50% Highest Density Region) and check if zero is within this interval. The test is positive if this is the case and negative otherwise (see Fig. 7.)

We run this test for 100 different networks and test values for $\iota = 509$ and its multiples. We consider we got the optimal $\iota$ when 50 of the 100 tested sender distributions result in a positive test. In our case this value was $\iota = 509 * 12 = 6108$, we chose a close-by prime number for the rest of the experiments $\iota = 6011$.

The burn-in period is the number of iterations needed until the Metropolis-Hastings walk mixes to reach the stationary distribution of the underlying Markov Chain. In our experiments, as the constraint problem explained in Sect 3.1 research was performed in parallel, we took as initial state the actual trace used for generating the network. Thus, in principle we could assume that from the beginning of the random walk the sampler is visiting states from the stationary distribution, and take the first $\mathcal{HS}$ visited as a valid sample. However, to avoid any influence of this a priori knowledge in our results, we chose a burn-in period of 8011 (bigger than $\iota$) that guarantees the first sample to be independent from the initial state.

The number of samples $N_{\mathrm{MH}}$ extracted from the sampler, can be arbitrary large to estimate the a-posterior distributions with a higher accuracy 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 had to set a value for the parameters $p_{\overline{cp}}$ and $p_{\mathrm{flip}}(a, b)$. The former defines the average percentage of non-compliant clients in the network and its choice is arbitrary. We decided to assign $p_{\overline{cp}} = 0.1$ such that the percentage of non-compliant clients using the network is small (as expected in a real network) but their presence in the network is non-negligible and we could study their effect on the analysis.

The probability $p_{\mathrm{flip}}(a, b)$ is also not crucial for the correctness of the sampler, but has an important role in the speed of mixing. The values we used in our experiments were chosen empirically to ensure fast mixing (i.e., that the sampler explores rapidly the full space and does not spend long times walking around small regions.) A study of the optimal values for $p_{\mathrm{flip}}(a, b)$ given $p_{\overline{cp}}$ is left as subject of future research.

The values for the parameters used in our experiments are summarised in Table 1. The network parameters were chosen to produce observations that can be analyzed. If we consider always a realistic mix network with at least $N_{\mathrm{mix}} = 10$ with threshold $t = 10$, when few messages (10 or 50) are sent most

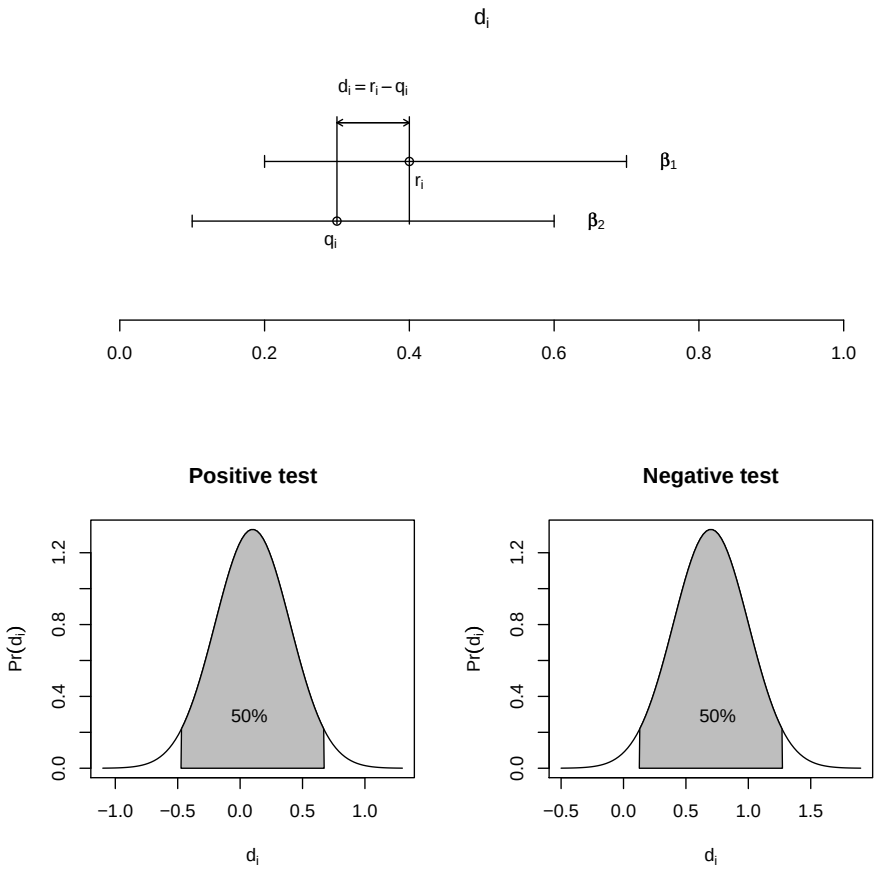



Figure 7: Construction of $d_i$ and result of the test

mixes would not flush and therefore there would be no flow of messages to analyze.

**Final test.** Once the parameters are set, we need to corroborate the correct functioning of our implementation, i.e., that the samples $\mathcal{HS}_j$ it returns actually follow the targeted probability distribution $\Pr[\mathcal{HS}]$.

Our test consists in 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:

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

The first element of the tuple is the inferred probability that $i_x$ corresponds to $o_y$ computed as in Eq. 5 from the result of the MH simulation. The second

| | Parameter | Value | | | |
|---|---|---|---|---|---|
| Network parameters | $N_{\text{msg}}$ | 10 | 50 | 100 | 1000 |
| | $N_{\text{mix}}$ | 3 | 3 | 10 | 10 |
| | t | 3 | 3 | 20 | 20 |
| | $[L_{\text{min}}, L_{\text{max}}]$ | [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 | | | |
| | $[L_{\text{min}_{\overline{cp}}}, L_{\text{max}_{\overline{cp}}}]$ | [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

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 network 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 this tuples whose second element is $I_{i_x \to o_y}(\text{trace}) = 1$ using the Beta function:

$$\alpha = \sum_{\substack{I_{i_x \to o_y} \\ \in \text{bin}(a,b)}} I_{i_x \to o_y}(\text{trace}) + 1\,,$$

$$\beta = \text{Len}(\text{bin}(a, b)) - \sum_{\substack{I_{i_x \to o_y} \\ \in \text{bin}(a,b)}} I_{i_x \to o_y}(\text{trace}) + 1\,,$$

$$p_{\text{empirical}}(a, b) \sim \text{Beta}(\alpha, \beta)\,.$$

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 simulation. 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 to describe the Beta function, thus, it accurately represents $p_{\text{empirical}}(a, b)$, and this result is very trustworthy. On the other hand, if few samples are used to compute the interval (if a bin contains few events), we obtain a poor estimation of $p_{\text{empirical}}(a, b)$ and the results based on it are rather meaningless. We conduct several experiments considering both the basic constraints and the full model (including non-compliant clients) in small and large networks, with the parameters described in Table 1.

Figure 8 shows the result of our evaluation when only basic constraints are considered both in the generation of the trace and in the analysis. The figure contains two graphs. The lower graph is just a histogram of the number of experiments per bin, $\text{Len}(\text{bin}(a, b))$. The upper graph represents with crosses $p_{\text{sampled}}(a, b)$, the mean of the bins, and the corresponding Bayesian confidence intervals $p_{\text{empirical}}(a, b)$ with vertical lines. We can see how most the crosses fall in the intervals, meaning that the sampler is providing $\mathcal{HS}_j$ according to the correct distribution. As expected, not all of the sampled probabilities $p_{\text{sampled}}(a, b)$ are within the intervals because we use a 95% confidence interval, thus approximately a 5% of them fall outside. The majority of messages entering in the network fall in bins with $p_{\text{sampled}} \in [0.07, 0.4]$. As we gather many samples of events with these probabilities the Bayesian confidence intervals are very small, indicating that the estimate we obtain is highly trustworthy.

It is noticeable that a fair amount of samples fall in the $p_{\text{sampled}} = 1$ bin. This denotes total certainty about the correspondence between an input and an output, which intuitively should not happen given the properties of mixes. These samples reflect two situations. On the one hand, they are generated when we select as target a message $i_x$ that starts in a path for which there exist no transition $Q$ leading to a new path in a suitable candidate $HS'$ and therefore stay invariable during the simulation. On the other hand, they are deterministic paths (explained in Sect. 3.2,) 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.

The following experiments were performed in scenarios where some of the clients behave in a non-compliant fashion. The result for $N_{\text{msg}} = 10$ messages is shown in Fig. 9(a). In this case, we observe many more events with $p_{\text{sampled}} = 1$. Again, we have some deterministic paths, but the increase is mainly due to the larger number of paths $P_x$ for which their links cannot be swapped resulting in suitable paths. Mainly, these are very long paths ($L_{x,\overline{cp}} >> L_{\max}$) chosen by non-compliant clients.

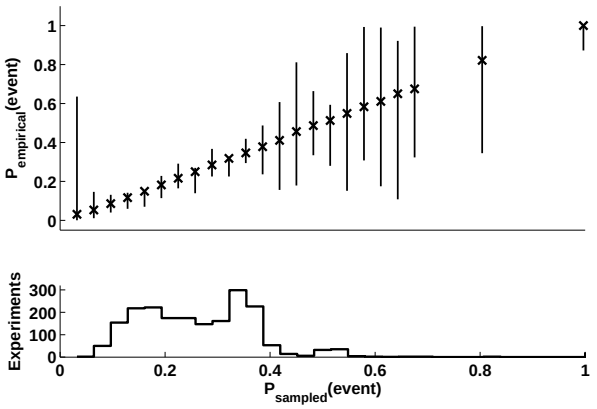


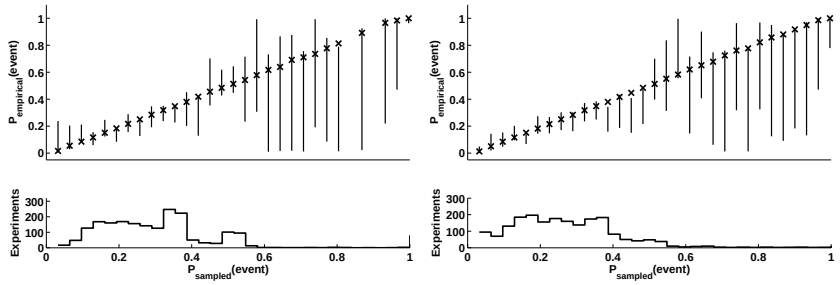

Figure 8: Results for the evaluation of an observation generated by 50 messages in a network with $N_{\text{mix}} = 3$ and $t = 3$, when all clients behave in a compliant way

The 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 samples represent the cases where paths would not have an alternative in the compliant case, but now can be caused with small probability by a non-compliant user. 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. Note that these events happen rarely and thus the number of samples falling in these bins is small resulting in large confidence intervals.

Figure 9(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, there are more messages where the attacker is less certain about the destination than when only few messages have entered in the network. Further, there are less 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.

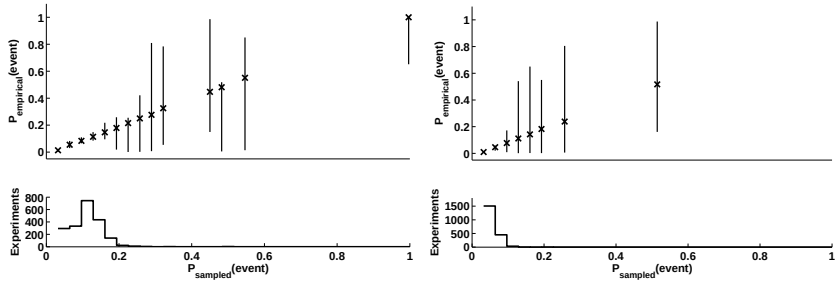
Finally, we tested the effectiveness of our sampler for longer observations (100 and 1000 messages in the network.) The results of the experiments are shown in Fig. 10. In these cases, our analysis finds that the mix network provides good anonymity for all messages. An attacker cannot link incoming and 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.

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




(a) $N_{\text{msg}} = 10$ messages (b) $N_{\text{msg}} = 50$ messages

Figure 9: Results for the evaluation of an observation of a network with $N_{\text{mix}} = 3$ and $t = 3$, when non-compliant clients are present




(a) $N_{\text{msg}} = 100$, $N_{\text{mix}} = 10$, $t = 20$ (b) $N_{\text{msg}} = 1000$, $N_{\text{mix}} = 10$, $t = 20$

Figure 10: Results for the evaluation of big networks

**Performance evaluation.** We now present an evaluation of the performance of our sampler in terms of LOC, RAM usage, and running times. We note that our implementation is not optimised for any of these metrics. We chose Python as developing language due to its ease for programming a prototype regardless of the existence of more suitable languages for fast implementations as would be C or C++. Our Metropolis-Hastings sampler is composed by 1443 LOC of Python, including the code associated to the evaluation.

The most remarkable measure we have taken in order to improve the performance of the sampler is the use of a "two-states" strategy for the proposal and acceptance/rejection of candidates $\mathcal{HS}'$. This strategy considers 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

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 |

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 |

total overhead, and significant ease of implementation.

As mentioned before, when implementing the sampler we did not optimise its RAM usage. Still, the memory requirements in order to run the sampler are well within the range of any commodity computer. Table 2 presents the RAM requirements for different sizes of the observation given by the parameters $N_{\text{mix}}$, $t$, and $N_{\text{msg}}$. The increasing need of RAM when the size of the network augments stems from the fact that the size of the observation $\mathcal{O}$, the $\mathcal{HS}$ and the population increases. Further, for each network analyzed, the samples $\mathcal{HS}_i$ are kept in RAM until the analysis finishes, 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 time it takes to analyse 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.

The increased running time when increasing the size of the network has two origins. On the one hand, the difference between the number of iterations $\iota$ considered. The larger the number of iterations, the more time the simulation needs to produce one sample. The second reason is that the timings include the collection of results for all the messages present in the system. The larger the number of messages, the longer it takes to store the results for all of them. If, as usual, the attacker considers only one target in the network, the time would be constant for any size of network. Although the time necessary to perform the analysis is already very convenient, it could be reduced considerably by performing several MH simulations in parallel for the same observation to accelerate the creation of samples $\mathcal{HS}_j$.

## 3.3 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 [11, 24, 58, 30], amongst which the most popular are the metrics based on Shannon entropy [26, 52]. 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 cell and $o_y$ an outgoing cell. 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 [52] 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.

Our 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 $\mathrm{Ct}(i_x \to o_y)$. 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 $\Psi_x$ [39]. We compute

the entropy $H(\Psi_x)$ of $n$ samples $\Psi_x$ from the posterior distribution:

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

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

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.

# 4 Bayesian Inference & Disclosure Attacks

Disclosure attacks were first proposed by Kesdogan *et al.* [35, 1] and later extended in the setting of the statistical disclosure attack [13, 21, 38, 40, 16, 60]. This line of work has been concerned with long term information leakage out of abstracted anonymity systems. It has been shown that unless all users benefit from near perfect anonymity for all messages they send, an adversary can build profiles of their correspondents over time, and use those to assist traffic analysis.

Sect. 2.3.3 is concerned with how to integrate such profiles into our model of a mix network. This section is concerned with both how to extract profiles and, in parallel, uncover who is talking with whom. We offer a generalisation of the Disclosure attack model of an anonymity system, as well as a Bayesian treatment of the inference problem associated with this attack.

## 4.1 The general Black-box model for anonymity systems

Long term attacks traditionally abstract the internal functioning of any anonymity system and represent it as an opaque router, effectively operating as a very large threshold mix. This model has its limitations, and some studies have attempted to extend it. In this section we first propose the Black-box model, the most flexible abstraction of an anonymity system so far, and base our Bayesian analysis on this model.

We start by proposing a 'forward' generative model describing how messages are generated and sent through the anonymity system. We then use Bayes rule to 'invert' the problem and perform inference on the unknown quantities. The broad outline of the generative model is depicted in Figure 11.

An anonymity system is abstracted as containing $N_{\text{user}}$ users that send $N_{\text{msg}}$ messages to each other. Each user is associated with a sending profile $\Psi_x$ describing how they select their correspondents when sending a message. We assume, in this work, that those profiles are simple multinomial distributions, that are sampled independently when a message is to be send to determine the receiver. We denote the collection of all sending profiles by $\Psi = \{\Psi_x | x = 1 \ldots N_{\text{user}}\}$.

A given sequence of users denoted by $\text{Sen}_1, \ldots, \text{Sen}_{N_{\text{msg}}}$ send a message while we observe the system. Using their sending profiles a corresponding sequence of

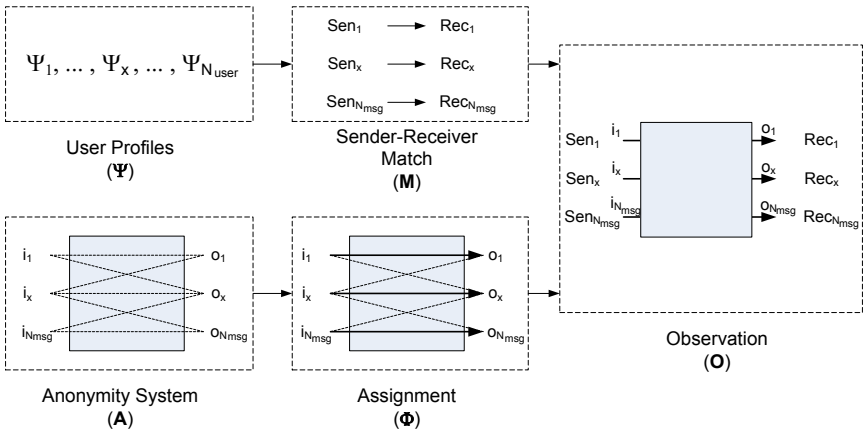



Figure 11: The generative model used for Bayesian inference in anonymous communications.

receivers $\mathrm{Rec}_1, \ldots, \mathrm{Rec}_{N_{\mathrm{msg}}}$ is selected to receive their messages. The probability of any receiver sequence is easy to compute. We denote this matching between senders and receivers as $\mathcal{M}$:

$$\Pr[\mathcal{M}|\Psi] = \prod_{x \in [1, N_{\mathrm{msg}}]} \Pr[\mathrm{Sen}_x \rightarrow \mathrm{Rec}_x|\Psi_x].$$

In parallel with the matching process where users choose their communication partners, an anonymity system $\mathcal{A}$ is used. This anonymity system is abstracted as a bipartite graph linking input messages $i_x$ with potential output messages $o_y$, regardless of the identity of their senders and receivers. We note that completeness of the bipartite graph is not required by the model. The edges of the bipartite graph are weighted with $w_{xy}$ that is simply the probability of the input message $i_x$ being output as $o_y$: $w_{xy} = \Pr[i_x \rightarrow o_y|\mathcal{A}]$.

This anonymity system $\mathcal{A}$ is used to determine a particular assignment of messages according to the weights $w_{xy}$. A single perfect matching on the bipartite graph described by $\mathcal{A}$ is selected to be the correspondence between inputs and outputs of the anonymity system for a particular run of the anonymity protocol. We call it the assignment of inputs to outputs and denote it by $\Phi$. The probability of the assignment $\Phi$ is easy to calculate, given the set of all individual assignments $(i_x \rightarrow o_x)$:

$$\Pr[\Phi|\mathcal{A}] = \prod_x \frac{\Pr[i_x \rightarrow o_x|\mathcal{A}]}{\sum_{\text{free } i_y} \Pr[i_y \rightarrow o_x|\mathcal{A}]}.$$

This is simply the probability of the matching given the anonymity system weights. By free $i_y$ we denote the set of sent messages $i$ that has not yet been assigned an output message $o$ as part of the match.

The assignment $\Phi$ of the anonymity system and the matching $\mathcal{M}$ of senders and receivers are composed to make up the observation of the adversary, that we denote as $\mathcal{O}$. An adversary observes messages from particular senders $\text{Sen}_x$ entering the anonymity as messages $i_x$, and on the other side messages $o_y$ exiting the network on their way to receivers $\text{Rec}_y$. No stochastic process takes place in this deterministic composition and therefore $\Pr[\mathcal{O}|\mathcal{M}, \Phi, \Psi, \mathcal{A}] = 1$.

Now that we have defined a full generative model for all the quantities of interest in the system, we turn our attention to the inference problem: the adversary observes $\mathcal{O}$ and knows about the anonymity system $\mathcal{A}$, but is ignorant about the profiles $\Psi$, the matching $\mathcal{M}$ and the assignment $\Phi$. We use Bayes theorem to calculate the probability $\Pr[\mathcal{M}, \Phi, \Psi|\mathcal{O}, \mathcal{A}]$. We start with the joint distribution and solve for it:

$$\begin{aligned}
\Pr[\mathcal{O}, \mathcal{M}, \Phi, \Psi|\mathcal{A}] &= \Pr[\mathcal{M}, \Phi, \Psi|\mathcal{O}, \mathcal{A}] \cdot \Pr[\mathcal{O}|\mathcal{A}] \\
\Pr[\mathcal{O}, \mathcal{M}, \Phi, \Psi|\mathcal{A}] &= \Pr[\mathcal{O}|\mathcal{M}, \Phi, \Psi, \mathcal{A}] && (\equiv 1) \\
&\quad \cdot \Pr[\mathcal{M}|\Phi, \Psi, \mathcal{A}] && (\equiv \Pr[\mathcal{M}|\Psi]) \\
&\quad \cdot \Pr[\Phi|\Psi, \mathcal{A}] && (\equiv \Pr[\Phi|\mathcal{A}]) \\
&\quad \cdot \Pr[\Psi|\mathcal{A}] \\
\Rightarrow \Pr[\mathcal{M}, \Phi, \Psi|\mathcal{O}, \mathcal{A}] &= \frac{\Pr[\mathcal{M}|\Psi] \Pr[\Phi|\mathcal{A}]}{\Pr[\mathcal{O}|\mathcal{A}] \equiv \mathcal{Z}} \Pr[\Psi|\mathcal{A}]
\end{aligned}$$

We have discussed how to calculate the probabilities $\Pr[\mathcal{M}|\Psi]$ and $\Pr[\Phi|\mathcal{A}]$. The quantity $\Pr[\Psi|\mathcal{A}] \equiv \Pr[\Psi]$ is the a-prior belief the attacker has about user profiles and it is independent from the chosen anonymity system $\mathcal{A}$. We consider throughout our analysis that all profiles are a-priori equally probable and reduce it to a constant $\Pr[\Psi] = c$. Taking into account those observations we conclude that the posterior probability sought is,

$$\Pr[\mathcal{M}, \Phi, \Psi|\mathcal{O}, \mathcal{A}] \sim \prod_{x \in [1, N_{\text{msg}}]} \Pr[\text{Sen}_x \to \text{Rec}_x|\Psi_x] \cdot \prod_x \frac{\Pr[i_x \to o_x|\mathcal{A}]}{\sum_{\text{free } i_y} \Pr[i_y \to o_x|\mathcal{A}]}$$

where we omit the constant normalising factor $\Pr[\mathcal{O}|\mathcal{A}]$ as it is very hard to calculate, which restricts the methods we can use to manipulate the a-posterior distribution.

It is computationally unfeasible to exhaustively enumerating the states of this distribution. Hence to calculate the marginals of interest such as profiles of users, or likely recipients of specific messages, we have to resort to sampling states from that distribution. Sampling directly is very hard (due to the inter-relation between the profiles, the matches and the assignments) hence Markov Chain Monte Carlo methods are used.

### 4.1.1 A Gibbs sampler for the Black-box model

Sampling states $(\mathcal{M}_j, \Phi_j, \Psi_j) \sim \Pr[\mathcal{M}, \Phi, \Psi|\mathcal{O}, \mathcal{A}]$ directly is hard, due to the complex interactions between the random variables. A Gibbs sampler significantly simplifies this process by only requiring us to sample from the marginal

distributions of the random variables sought. Given an arbitrary initial state $(\Phi_0, \Psi_0)$ we can perform $\iota$ iterations of the Gibbs algorithm as follows:

$$\begin{aligned} &\text{for } j := 1 \dots \iota : \\ \Phi_j, \mathcal{M}_j &\sim \Pr[\Phi, \mathcal{M} | \Psi_{j-1}, \mathcal{O}, \mathcal{A}] \\ \Psi_j &\sim \Pr[\Psi | \Phi_j, \mathcal{M}_j, \mathcal{O}, \mathcal{A}] \quad . \end{aligned}$$

Each of these marginal probabilities distributions is easy to sample:

- The distribution of assignments $\Pr[\Phi, \mathcal{M} | \Psi_{j-1}, \mathcal{O}, \mathcal{A}]$ is subtle to sample directly. Each message assignment $i_x \to o_x$ has to be sampled, taking into account that some message assignments are already taken by the time input message $i_x$ is considered. For each input message $i_x$ we sample an assignment $o_y$ according to the distribution:

$$\begin{aligned} i_x \to o_y \sim &\Pr[i_x \to o_y | \text{free } o_y, \forall_{\text{assigned } o_v} i_v \to o_v, \mathcal{A}, \Psi] \\ = &\frac{\Pr[i_x \to o_y | \mathcal{A}] \cdot \Pr[\text{Sen}_x \to \text{Rec}_y | \Psi_x]}{\sum_{\text{free } o_y} \Pr[i_x \to o_y | \mathcal{A}] \cdot \Pr[\text{Sen}_x \to \text{Rec}_y | \Psi_x]} \quad . \end{aligned}$$

  For complex anonymity systems $\mathcal{A}$, this algorithm might return only partial matches, when at some point an input message $i_x$ has no unassigned candidate output message $o_y$ left. Since we are only interested in perfect matchings, where all input messages are matched with different output messages, we reject such partial states and re-start the sampling of the assignment until a valid perfect matching is returned. This is effectively a variant of rejection sampling, to sample valid assignments.

  The matchings between senders and receivers are uniquely determined by the assignments and the observations, so we can update them directly without any need for sampling, and regardless of the profiles (i.e. $\mathcal{M}_j = f(\Psi_j, \mathcal{O})$).

- The distribution of profiles $\Pr[\Psi | \Phi_j, \mathcal{M}_j, \mathcal{O}, \mathcal{A}]$ is straightforward to sample given the matching $\mathcal{M}_j$ and assuming that individual profiles $\Psi_x$ are multinomial distributions.

  We note that the Dirichlet distribution is a conjugate prior of the multinomial distribution, and we use it to sample profiles for each user. We denote as $\Psi_x = (\Pr[\text{Sen}_x \to \text{Rec}_1], \dots, \Pr[\text{Sen}_x \to \text{Rec}_{N_{\text{user}}}])$ the multinomial profile of user $\text{Sen}_x$. We also define a function that counts the number of times a user $\text{Sen}_x$ is observed sending a message to user $\text{Rec}_y$ in the match $\mathcal{M}$, and denote it as $\text{Ct}_{\mathcal{M}}(\text{Sen}_x \to \text{Rec}_y)$. Sampling profiles $(\Psi_1, \dots, \Psi_{N_{\text{user}}}) \sim \Pr[\Psi | \mathcal{M}]$ involves sampling independently each sender's profile $\Psi_x$ separately from a Dirichlet distribution with the following parameters:

$$\Psi_x \sim \text{Dirichlet}(\text{Ct}_{\mathcal{M}}(\text{Sen}_x \to \text{Rec}_1)+1, \dots, \text{Ct}_{\mathcal{M}}(\text{Sen}_x \to \text{Rec}_{N_{\text{user}}})+1) \quad .$$

If the anonymity system $\mathcal{A}$ describes a simple bipartite graph, the rejection sampling algorithm described can be applied to sample assignments $i_x \to o_x$ for all messages. When this variant of rejection sampling becomes expensive, due to a large number of rejections, a Metropolis-Hastings [10] based algorithm can be used to sample perfect matchings on the bipartite graph according to the distribution $\Pr[\Phi, \mathcal{M}|\Psi_{j-1}, \mathcal{O}, \mathcal{A}]$.

The Gibbs sampler can be run multiple times to extract multiple samples from the a-posterior distribution $\Pr[\mathcal{M}, \Phi, \Psi|\mathcal{O}, \mathcal{A}]$. Instead of restarting the algorithm at an arbitrary state $(\mathcal{M}_0, \Phi_0, \Psi_0)$, it is best to set the starting state to the last extracted sample, that is likely to be within the typical set of the distribution. This speeds up convergence to the target distribution.

## 4.2 A computationally simple Red-Blue model

After the PMDA [60] it has become dogma that sender profiles have to be co-estimated simultaneously with the assignments, and our Bayesian analysis so far reflects this approach. Senders are associated with mutinomial profiles with which they choose specific correspondents. We sample these profiles using the Dirichlet distribution, and use them to directly sample weighted perfect assignments in the anonymity system. The output of the algorithm is a set of samples of the hidden state, that allows the adversary to estimate the marginal distributions of specific senders sending to specific receivers.

We note that this approach is very generic, and might go beyond the day to day needs of a real-world adversary. An adversary is likely to be interested in particular target senders or receivers, and might want to answer the question: "who has sent this message to Bob?" or "who is friends with receiver Bob?". We present the Red-Blue model to answer such questions, which is much simpler, both mathematically and computationally, than the generic model presented so far.

Consider that the adversary chooses a target receiver Bob (that we call "Red"), while ignoring the exact identity of all other receivers and simply tagging them as "Blue". The profiles $\Psi_x$ of each sender can be collapsed into a simple binomial distribution describing the probability sender $x$ sends to Red or to Blue. It holds that:

$$\Pr[\text{Sen}_x \to \text{Red}|\Psi_x] + \Pr[\text{Sen}_x \to \text{Blue}|\Psi_x] = 1. \tag{7}$$

Matchings $\mathcal{M}$ map each observed sender of a message to a receiver class, either Red or Blue. Given the profiles $\Psi$ the probability of a particular match $\mathcal{M}$ is:

$$\Pr[\mathcal{M}|\Psi] = \prod \Pr[\text{Sen}_x \to \text{Red / Blue}|\Psi_x]$$

The real advantage of the Red-Blue model is that different assignments $\Phi$ now belong to equivalence classes, since all Red or Blue receivers are considered indistinguishable from each other. In this model the assignment bipartite graph can be divided into two sub-graphs: the sub-graph $\Phi_R$ contains all edges ending on the Red receiver (as she can receive more than one message in a mixing

round), while the sub-graph $\Phi_B$ contains all edges ending on a Blue receiver. We note that these sub-graphs are complementary and any of them uniquely defines the other. The probability of each $\Phi$ can then be calculated as:

$$\begin{aligned}\Pr[\Phi|\mathcal{A}] &= \sum_{\forall \Phi_B} \Pr[\Phi_B, \Phi_R|\mathcal{A}] = \\ &= \sum_{\forall \Phi_B} \Pr[\Phi_B|\Phi_R, \mathcal{A}] \cdot \Pr[\Phi_R|\mathcal{A}] = \\ &= \Pr[\Phi_R|\mathcal{A}] \cdot \sum_{\forall \Phi_B} \Pr[\Phi_B|\Phi_R, \mathcal{A}] = \\ &= \Pr[\Phi_R|\mathcal{A}]\end{aligned}$$

The probability of an assignment in an equivalence class defined by the assignment to Red receivers, only depends on $\Phi_R$ describing this assignment. The probability of assignment $\Phi_R$ can be calculated analytically as:

$$\Pr[\Phi_R|\mathcal{A}] = \prod_{x \in \Phi_R} \frac{\Pr[i_x \to o_x]}{\sum_{\text{free } i_j} \Pr[i_j \to o_x]}.$$

The assignment $\Phi_R$ must be a sub-graph of at least one perfect matching on the anonymity system $\mathcal{A}$, otherwise the probability becomes $\Pr[\Phi|\mathcal{A}] = 0$. As for the full model the probability of all the hidden quantities given the observation is:

$$\Pr[\mathcal{M}, \Phi, \Psi|\mathcal{O}, \mathcal{A}] = \frac{\Pr[\mathcal{M}|\Psi] \Pr[\Phi_R|\mathcal{A}]}{\Pr[\mathcal{O}|\mathcal{A}] \equiv \mathcal{Z}} \Pr[\Psi|\mathcal{A}] \tag{8}$$

The a-prior probability over profiles $\Pr[\Psi|\mathcal{A}]$ is simply a prior probability over parameters of a binomial distribution. Each profile can be distributed as $\Pr[\Psi_x|\mathcal{A}] = \text{Beta}(1,1)$ if nothing is to be assumed about the sender's $x$ relationship with the Red receiver.

In practice a prior distribution $\Pr[\Psi_x|\mathcal{A}] = \text{Beta}(1,1)$ is too general, and best results are achieved by using a more skewed distribution as for example $\text{Beta}(1/100, 1/100)$. This reflects the fact that social ties are a-prior either strong or non existent. Given enough evidence the impact of this choice of prior fades quickly away.

#### 4.2.1 A Gibbs sampler for the Red-Blue model

Implementing a Gibbs sampler for the Red-Blue model is very simple. The objective of the algorithms is, as for the general model, to produce samples of profiles $(\Psi_j)$, assignments and matches $(\Phi_j, \mathcal{M}_j)$ distributed according to the Bayesian a-posterior distribution $\Pr[\mathcal{M}, \Phi, \Psi|\mathcal{O}, \mathcal{A}]$ described by eq. 8.

The Gibbs algorithm starts from an arbitrary state $(\Psi_0, \Phi_0)$ and iteratively samples new marginal values for the profiles $(\Phi_j, \mathcal{M}_j \sim \Pr[\Phi, \mathcal{M}|\Psi_{j-1}, \mathcal{O}, \mathcal{A}])$ and the valid assignments $(\Psi_j \sim \Pr[\Psi|\mathcal{M}_j, \Phi_j, \mathcal{O}, \mathcal{A}])$. The full matchings are a deterministic function of the assignments and the observations, so we can update them directly without any need for sampling (i.e. $\mathcal{M}_j = f(\Psi_j, \mathcal{O})$).

As for the general Gibbs sampler, sampling from the desired marginal distributions can be done directly. Furthermore the Red-Blue model introduces some simplifications that speed up inference:

- **Sampling assignments.** Sampling assignments of senders to Red nodes (i.e. $\Phi_{Rj}, \mathcal{M}_j \sim \Pr[\Phi, \mathcal{M}|\Psi_{j-1}, \mathcal{O}, \mathcal{A}]$) can be performed by adapting the rejection sampling algorithm presented for the general model. The key modification is that only assignments to Red receivers are of interest, and only an arbitrary assignment to blue receivers is required (to ensure such an assignment exists). This time for each Red output messages $o_x$ we sample an input message $i_x$ according to the distribution:

$$\begin{aligned} i_x \to o_y \sim & \Pr[i_x \to o_y | \text{free } i_x, \forall_{\text{assigned } i_v} i_v \to o_v, \mathcal{A}, \Psi] \\ = & \frac{\Pr[i_x \to o_y|\mathcal{A}] \cdot \Pr[\text{Sen}_x \to \text{Red}|\Psi_x]}{\sum_{\text{free } i_j} \Pr[i_j \to o_y|\mathcal{A}] \cdot \Pr[\text{Sen}_j \to \text{Red}|\Psi_x]} \end{aligned}$$

- **Sampling profiles.** Sampling a profile $\Psi_j \sim \Pr[\Psi|\mathcal{M}_j, \Phi_j, \mathcal{O}, \mathcal{A}]$ for every user $x$ simply involves drawing a sample from a Beta distribution with parameters related to the number of links to Blue and Red receivers. To be formal we define a function $\text{Ct}_{\mathcal{M}}(\text{Sen}_x \to \text{Red, Blue})$ that counts the number of messages in a match that a user sender $x$ sends to a Red or Blue receiver. The profile of user $x$ is then sampled as:

$$\Psi_x \sim \text{Beta}(\text{Ct}_{\mathcal{M}}(\text{Sen}_x \to \text{Blue}) + 1, \text{Ct}_{\mathcal{M}}(\text{Sen}_x \to \text{Red}) + 1)$$

  This yields a binomial parameter that is the profile of user $x$, describing the probability they send a message to a Red target user.

The cost of each iteration is proportional to sampling $N_{\text{user}}$ Beta distributions, and sample from the distribution of senders of each of the Red messages. Both the sampling of profiles, and the sampling of assignments can be performed in parallel, depending on the topology. In case a large number of samples are needed multiple Gibbs samplers can be run on different cores or different computers to produce them.

## 4.3 Evaluation

The Red-Blue model for inferring user profiles and assignments was evaluated against synthetic anonymized communication traces, to test its effectiveness. The communication traces include messages sent by up to 1000 senders to up to 1000 receivers. Each sender is assigned 5 contacts at random, to whom they send messages with equal probability. Messages are anonymized in discrete rounds using a threshold mix that gathers 100 messages before sending them to their receivers as a batch.

The generation of communication patterns was peculiar to ensure a balance between inferring the communications of a target user (as in the traditional disclosure, hitting set and statistical disclosure attacks) to a designated Red

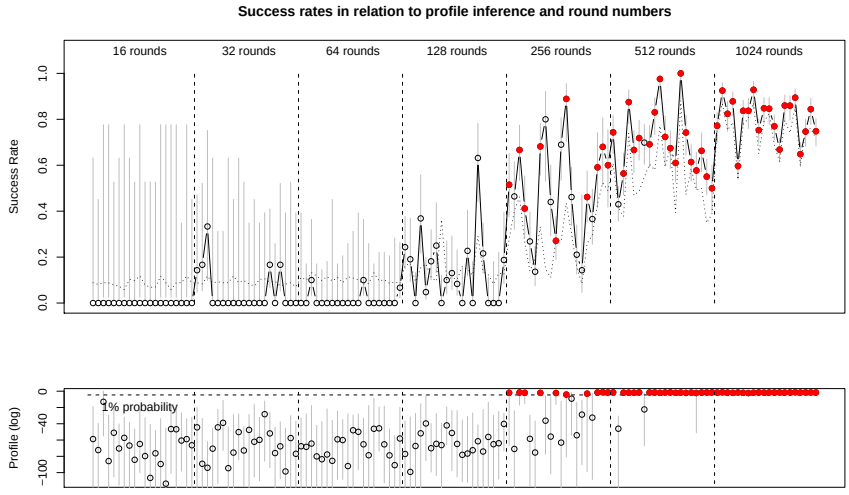



Figure 12: Performance of the Red-Blue model in assigning senders to the target red receiver, as a function of the number of rounds observed. Twenty sample experiments are used per box plot.

receiver, as well as to gain enough information about other users to build helpful profiles for them. A target sender was included in 20% of the rounds, and the Red node was chosen to be one of their friends. A sequence of experiments were performed to assess the accuracy of the attack after observing an increasing number of rounds of communication.

The aim of each experiment is to use the samples returned by a Gibbs sampler implementing the Red-Blue model to guess the sender of each message that arrives at a designated Red receiver. The optimal Bayes criterion [8] is used to select the candidate sender of each Red message: the sender with the highest a-posterior probability is chosen as the best candidate. This probability is estimated by counting the number of times each user were the sender of a target Red message in the samples returned by the Gibbs algorithm. The Bayesian probability of error, i.e. the probability another sender is responsible for the Red message, is also extracted, as a measure of the certainty of each of these "best guesses". For each experiment the Gibbs sampler was used to extract 200 samples, using 100 iterations of the Gibbs algorithm each. The first 5 samples were discarded, to ensure stability is reached before drawing any inferences.

A summary of the results for each experiment is presented in Figure 12. The top graph illustrates the fraction of correct guesses per experiment (on the $x$ axis – we selected 20 random experiments to display per round number) grouped by the number of rounds of communication observed (16, 32, 64, 128, 256, 512 and

1024). For each experiment the fraction of correctly identified senders is marked by a circle, along with its 90% confidence interval. The dashed line of the same graph represents the prediction of error we get from the Bayesian probability of error. The bottom graph on Figure 12 illustrates on a logarithmic scale the inferred probability assigned to the Red node for the target sender, for each of the experiments. The experiments for which a high value of this probability are inferred (median greater than 1%) are marked by a solid red circle on both graphs. The 50% confidence interval over the profile parameter is also plotted.

Some key conclusions emerge from the experiments illustrated on Figure 12:

- The key trend we observe is, as expected, that the longer the observation in terms of rounds, the better the attack. Within 1024 rounds we expect the target sender to have sent about 40 messages to the designated red target. Yet, the communication is traced to them on average 80% of the cases with high certainty. Even when only 256 rounds are observed the correct assignment is guessed in about 50% of the time.

- The quality of the inference when it comes to the correspondence between messages, senders and receivers, is intimately linked to the quality of the profile inference. The solid red circles mark experiments that concluded that the media value for the probability the target sender is friends with the target Red receiver is high (greater than 1%). We observe that these experiments are linked to high success rates when it comes to linking individual messages to the target sender. We also observe the converse: insufficient data leads to poor profiles, that in turn lead to poor predictions about communication relationships.

- The probability of error estimates (represented on the top graph by a dotted line) predict well the success rate of the experiments. The predicted error rate systematically falls within the 90% confidence interval of the estimated error rate. This shows that the Red-Blue model is a good representation of the process that generated the traces and thus the estimates coincide with the actual observed error rate, on average. This is due to the very generic model for Red-Blue profiles that represent reality accurately after a few rounds. Yet, when few rounds are observed the a-prior distribution of profiles dominates the inference, and affects the error estimates.

A key question is how the results from the Red-Blue model compare with traditional traffic analysis attacks, like the SDA [40], the NSDA [60] or the PMDA [60]. The SDA attack simply uses first order frequencies to guess the profiles of senders. It is fast but inaccurate. The normalised SDA (NSDA) constructs a traffic matrix from senders to receivers, that is normalised to be doubly stochastic. The operation is as fast as matrix multiplication, and yields very good results. The PMDA finds perfect matchings between senders and receivers based on a rough profile extraction step – it is quite accurate but slow.

Figure 13 illustrates the relative performances of the different attacks compared with the Red-Blue model proposed. We observe that the inference based

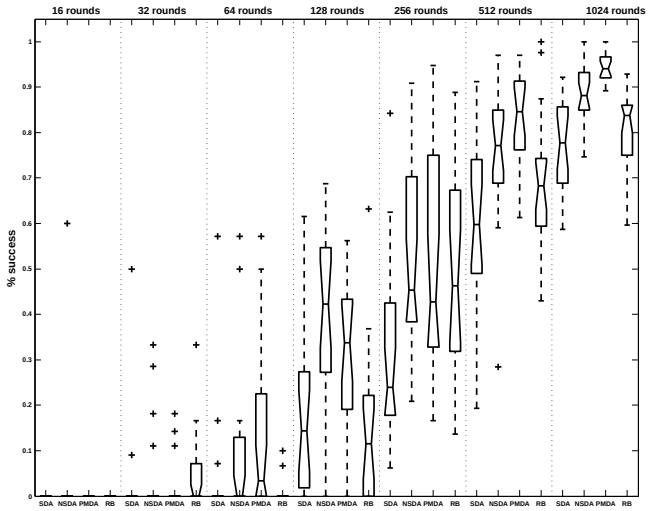



Figure 13: Performance of the Red-Blue inference model compared to the SDA, NSDA and PMDA.

technique is quite competitive, against the SDA, but performs worse than the NSDA and PMDA in most settings. This is due to our strategy for extracting best estimates for the senders: we use the output samples to chose the sender with highest marginal probability instead of extracting a full match with the maximal marginal probability. In that sense applying an algorithm to find the maximal perfect matching based on the marginal probabilities output by the RB attacks should produce much better results.

Despite the lower success rate inference based techniques can be advantageous. Their key strength is the certainty that no systematic bias has been introduced by reusing data twice, as reported in [27, 60], and the tangible and reliable error estimate they output. A traffic analyst is thus able to judge the quality of the inference to guide them operationally.

A second important advantage is the ability to infer who is the "second most likely" receiver, compute anonymity metrics, or other arbitrary statements on the a-posterior probability distribution of profiles and assignments. This can be done efficiently simply using the samples output by the Gibbs algorithm. Furthermore the correct probabilities of error can be associated with those probabilistic statements.

## 4.4 Discussion & future directions

The Bayesian treatment of long term attacks against anonymity systems is promising, but still at its infancy. We foresee some key theoretical as well

as implementation steps to move the state of the art forward.

- **Bipartite weighted anonymity set**. The Black-box model as well as the Red-Blue model proposed represent an observation from an anonymity system as a generic weighted bipartite graph, linking senders with receivers. Our experiments, on the other hand, only considered anonymity systems working in discrete rounds, forming full bipartite sub-graphs with a number of senders equal to the batch size. This is a limitation of our sampler implementation, that could be extended to deal with the general case of any bipartite weighted network.

  While in theory this modification is straightforward, in practice it is harder to sample directly matchings from arbitrary bipartite graphs. The rejection sampling algorithm suggested can be inefficient, since it might use links that are not part of a perfect matching, forcing multiple aborts. It might be wise to first prune the assignment graph from such edges using techniques from the constrain satisfaction literature such as Regin's algorithm [50].

- **Profile models**. The a-prior model for user profiles is very generic, meaning that it can represent, and thus learn, any multinomial distribution of receivers per sender. While being generic more information could be incorporated if it is established that the profile belongs to a social network (with some standard characteristics like degree, clustering etc). Traditional hitting set as well as disclosure attacks make extensive use of the number of friends of a target sender to be applicable at all, whereas the presented approaches do not require such information. Yet, adding related constraints would yield better results.

- **Learning social networks**. It has been an open problem in the literature how to incorporate known information about communication patterns to help the inference of unknown communication patterns, and some ad-hoc techniques were presented to combine social network information to de-anonymize traces, along with a discussion of systematic errors introduced [27]. The sampling techniques presented in this work can be straightforwardly modified to incorporate known correspondences between senders and receivers: the Gibbs sampler is modified to only sample valid assignments that contain the known matches. These known assignments, far from being useless, drive the sampling of profiles (as part of the Gibbs sampling) leading to higher quality profiles, which in turn become higher quality assignments for the unknown messages.

- **Beyond communications.** Both models presented are very generic and apply to attempts to anonymize traces that are not communications. As long as a system has users with multinomial preferences, that are expressed and anonymized in an arbitrary manner (as long as there is one expressed preference per observed action), our algorithms are applicable to de-anonymize the preferences and extract user profiles. This problem

has recently received considerable attention though de-anonymization algorithms applied to the NetFlix database [44].

# 5 Conclusions

Each proposed mix system is slightly different from others, and our model has to still be extended to deal with different mixing strategies [54, 46], dummy traffic [19, 25, 46] 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 [57], such as Tor [28]. 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 [42] 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, without resorting to heuristics.

The second contribution of this work is Vida, the first truly general model for abstracting any anonymity system, in the long term, to perform de-anonymization attacks. Users and their preferences are modelled in the most generic way, using multinomial profile, eliminating the need to know the number of contacts each sender has. Instead of abstracting an anonymity system as a single threshold mix, or even pool mix, an arbitrary weighted mapping of input to output messages can be used. We show that the model performs well when it comes to guessing who is talking to whom, as well as guessing the profiles of senders. The Vida Red-Blue model focuses on the need the working traffic analyst has to infer patterns of communications to specific targets – it has the potential to be implemented efficiently and parallelized aggressively.

The third contribution is methodological, and might be even more significant than the specific models. We demonstrate that probabilistic modelling, Bayesian inference, and the associated conceptual toolkit relating to Monte Carlo Markov Chain sampling is an appropriate framework upon which to build traffic analysis attacks. It ensures that information is used properly avoiding over fitting or systematic biases; it provides a clear framework to perform the analysis starting with the definition of a probabilistic model, that is inverted and sampled to estimate quantities of interest; it provides good and clear estimates of error, as well as the ability to answer arbitrary questions about the hidden state with a

clear probability statement. These qualities are in sharp contrast with the state of the art in traffic analysis, that provides ah-hoc best guesses of very specific quantities, with a separate analysis to establish their accuracy based on labeled data – something that the traffic analyst does not have on the ground.

We hope this work is the start of an exploration of the applicability of inference techniques to problems in traffic analysis – that will eventually outperform established techniques. Some clear future directions include the definition of better user models, the analysis of the internals of anonymity systems, as well as a better integration of prior information and learning. The inference approach leans itself well to be extended to encompass these problems, that have in the past been a thorn on the side of traffic analysis techniques.

# References


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

[2] Dakshi Agrawal, Dogan Kesdogan, and Stefan Penz. Probabilistic Treatment of MIXes to Hamper Traffic Analysis. In *Proceedings of the 2003 IEEE Symposium on Security and Privacy*, pages 16–27, May 2003.

[3] Oliver Berthold, Andreas Pfitzmann, and Ronny Standtke. The disadvantages of free MIX routes and how to overcome them. In H. Federrath, editor, *Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, pages 30–45. Springer-Verlag, LNCS 2009, July 2000.

[4] Lucas Bordeaux, Youssef Hamadi, and Lintao Zhang. Propositional satisfiability and constraint programming: A comparative survey. *ACM Comput. Surv.*, 38(4), 2006.

[5] Nikita Borisov, George Danezis, Prateek Mittal, and Parisa Tabriz. Denial of service or denial of security? How attacks on reliability can compromise anonymity. In *Proceedings of CCS 2007*, October 2007.

[6] Jan Camenisch and Anna Lysyanskaya. A formal treatment of onion routing. In Victor Shoup, editor, *Proceedings of CRYPTO 2005*, pages 169–187. Springer-Verlag, LNCS 3621, August 2005.

[7] K. Chatzikokolakis, C. Palamidessi, and P. Panangaden. Anonymity Protocols as Noisy Channels? *Proc. 2nd Symposium on Trustworthy Global Computing, LNCS. Springer*, 2006.

[8] Konstantinos Chatzikokolakis, Catuscia Palamidessi, and Prakash Panangaden. Probability of error in information-hiding protocols. In *Proceedings of the 20th IEEE Computer Security Foundations Symposium (CSF20)*, 2007.

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

[10] Siddhartha Chib and Edward Greenberg. Understanding the metropolis-hastings algorithm. *The American Statistician*, 49(4):327–335, 1995.

[11] 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.

[12] George Danezis. Mix-networks with restricted routes. In Roger Dingledine, editor, *Proceedings of Privacy Enhancing Technologies workshop (PET 2003)*, pages 1–17. Springer-Verlag, LNCS 2760, March 2003.

[13] 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.

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

[15] 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.

[16] 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.

[17] 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.

[18] George Danezis and Ben Laurie. Minx: A simple and efficient anonymous packet format. In *Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2004)*, Washington, DC, USA, October 2004.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

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

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

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

[31] A. Gelman, J. B. Carlin, H. S. Stern, and D. B. Rubin. *Bayesian Data Analysis*. Chapman and Hall, London, 1995.

[32] Andrew Gelman, John B. Carlin, Hal S. Stern, and Donald B. Rubin. *Bayesian Data Analysis, Second Edition*. Chapman & Hall/CRC, July 2003.

[33] 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.

[34] 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.

[35] 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.

[36] Dogan Kesdogan, Jan Egner, and Roland Büschkes. Stop-and-go MIXes: Providing probabilistic anonymity in an open system. In *Proceedings of Information Hiding Workshop (IH 1998)*. Springer-Verlag, LNCS 1525, 1998.

[37] 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.

[38] 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.

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

[40] 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.

[41] Bodo Möller. Provably secure public-key encryption for length-preserving chaumian mixes. In *Proceedings of CT-RSA 2003*. Springer-Verlag, LNCS 2612, April 2003.

[42] Steven J. Murdoch and Piotr Zielinski. Sampled traffic analysis by internet-exchange-level adversaries. In *Privacy Enhancing Technologies*, pages 167–183, 2007.

[43] Arjun Nambiar and Matthew Wright. Salsa: A structured approach to large-scale anonymity. In *Proceedings of CCS 2006*, October 2006.

[44] Arvind Narayanan and Vitaly Shmatikov. Robust de-anonymization of large sparse datasets. In *IEEE Symposium on Security and Privacy*, pages 111–125. IEEE Computer Society, 2008.

[45] Richard E. Newman, Vipan R. Nalla, and Ira S. Moskowitz. Anonymity and covert channels in simple timed mix-firewalls. In *Proceedings of Privacy Enhancing Technologies workshop (PET 2004)*, volume 3424 of *LNCS*, pages 1–16, May 2004.

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

[47] Lasse Øverlier and Paul Syverson. Locating hidden servers. In *Proceedings of the 2006 IEEE Symposium on Security and Privacy*. IEEE CS, May 2006.

[48] Michael A. Padlipsky, David W. Snow, and Paul A. Karger. Limitations of end-to-end encryption in secure computer networks. Technical Report ESD-TR-78-158, The MITRE Corporation: Bedford MA, HQ Electronic Systems Division, Hanscom AFB, MA, August 1978.

[49] Jean-François Raymond. Traffic Analysis: Protocols, Attacks, Design Issues, and Open Problems. In H. Federrath, editor, *Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, pages 10–29. Springer-Verlag, LNCS 2009, July 2000.

[50] Jean-Charles Régin. A filtering algorithm for constraints of difference in csps. In *AAAI*, pages 362–367, 1994.

[51] Marc Rennhard and Bernhard Plattner. Introducing MorphMix: Peer-to-Peer based Anonymous Internet Usage with Collusion Detection. In *Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2002)*, Washington, DC, USA, November 2002.

[52] 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.

[53] Andrei Serjantov, Roger Dingledine, and Paul Syverson. From a trickle to a flood: Active attacks on several mix types. In Fabien Petitcolas, editor, *Proceedings of Information Hiding Workshop (IH 2002)*. Springer-Verlag, LNCS 2578, October 2002.

[54] 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.

[55] Eric Shimshock, Matt Staats, and Nick Hopper. Breaking and provably fixing minx. In Nikita Borisov and Ian Goldberg, editors, *Proceedings of the Eighth International Symposium on Privacy Enhancing Technologies (PETS 2008)*, pages 99–114, Leuven, Belgium, July 2008. Springer.

[56] Vitaly Shmatikov and Ming-Hsui Wang. Measuring relationship anonymity in mix networks. In *Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2006)*, October 2006.

[57] 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.

[58] 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.

[59] Gergely Tóth, Zoltán Hornák, and Ferenc Vajda. Measuring anonymity revisited. In Sanna Liimatainen and Teemupekka Virtanen, editors, *Proceedings of the Ninth Nordic Workshop on Secure IT Systems*, pages 85–90, Espoo, Finland, November 2004.

[60] 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.

[61] Carmela Troncoso, Benedikt Gierlichs, Bart Preneel, and Ingrid Verbauwhede. Perfect matching statistical disclosure attacks. In Nikita Borisov and Ian Goldberg, editors, *Proceedings of the Eighth International Symposium on Privacy Enhancing Technologies (PETS 2008)*, pages 2–23, Leuven, Belgium, July 2008. Springer.

[62] 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.

# A Notation

Table 4: Notation I

| Symbol | Meaning |
|---|---|
| $\mathcal{A}$ | Generic anonymity system |
| $\alpha \to \beta$ | $\alpha$ sends a message to $\beta$ |
| $\mathcal{C}$ | Set of constraints imposed by the system |
| $\overline{\mathcal{C}}$ | Set of constraints for non compliant users |
| $\mathrm{Ct}(\alpha \to \beta)$ | number of times $\alpha \to \beta$ appears in the set of samples |
| $cp$ | Path compliant with system standard specifications |
| $\overline{cp}$ | Path *non* compliant with system standard specifications |
| $\Phi$ | Assignment of inputs to outputs |
| $\mathcal{HS}$ | Hidden State |
| $i_x$ | Input message $x$ |
| $\iota$ | Number of iterations in the Metropolis-Hasting simulation between the collection of two samples |
| $L_x$ | Length of path $P_x$ |
| $\mathrm{Lab}_x$ | Label of path x ($\mathrm{Lab}_x = cp$ or $\mathrm{Lab}_x = \overline{cp}$) |
| $M_x$ | Sequence of mixes belonging to path $P_x$ |
| $\mathcal{M}$ | Matching between senders and receivers |
| $N_{\mathrm{mix}}$ | Number of mixes in the network |
| $N_{\mathrm{msg}}$ | Number of messages in the system |
| $N_{\mathrm{user}}$ | Number of users in the population |
| $N_{\mathrm{det}}$ | Number of deterministic paths in an observation |
| $\mathcal{O}$ | Observation |
| $N_{\mathrm{MH}}$ | Number of samples gathered in the MH simulation |
| $o_y$ | Output message $y$ |
| $o_{\overline{y}}$ | Output message $y$ that has not been sampled |
| $\mathcal{P}$ | The set of all paths in the observation |
| $P_x$ | Path $x$, starting at input message $i_x$ |
| $\Pr[\mathcal{HS}]$ | Abbreviation for $\Pr[\mathcal{HS}\|\mathcal{O},\mathcal{C}] = \Pr[\mathcal{P}\|\mathcal{C}]$ |
| $p_{\overline{cp}}$ | Probability of non-compliant client |
| $p_{\mathrm{flip}}(a,b)$ | Probability of changing the label of a path from $\mathrm{Lab}_x = a$ to $\mathrm{Lab}_x = b$ |

# B A typical small trace

Table 5: Notation II

| Symbol | Meaning |
|---|---|
| $p_{\text{sampled}}(a, b)$ | Mean of the probabilities given the MH-simulation for events with probability $a \leq \Pr[event] < b$ |
| $p_{\text{empirical}}(a, b)$ | Mean of the probabilities given the observations for events with probability $a \leq \Pr[event] < b$ |
| $\pi$ | Permutation of relation between input and output messages |
| $\Psi_x$ | Sending profile of user $\text{Sen}_x$ |
| $\text{Rec}_y$ | Receiver of output message $o_y$ |
| $\text{Sen}_x$ | Sender of input message $i_x$ |
| $T_0$, $T_{\max}$ | Beginning and end of the observation time |
| $t$ | Threshold of mixes |
| unf | Subscript related to paths ending in an unflushed mix |

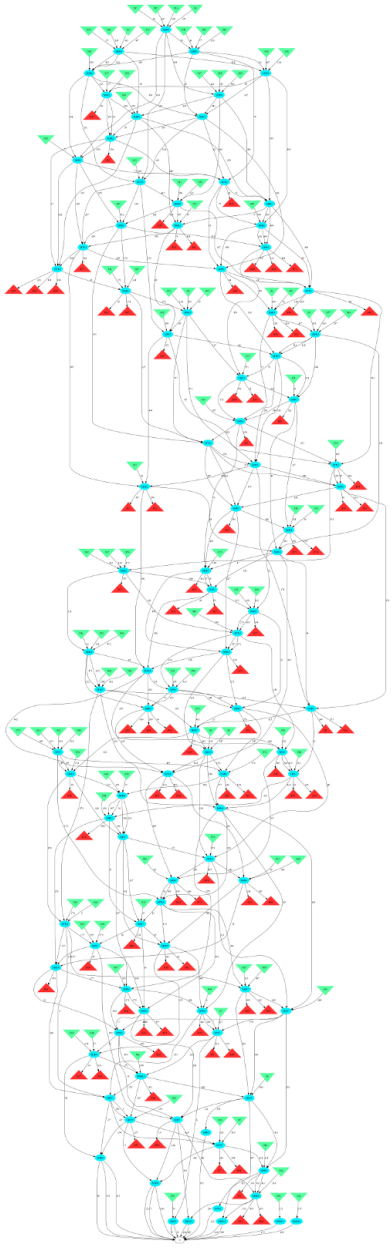


Figure 14: A typical non-toy observation