Universitat Autònoma de Barcelona

Anonymity and privacy on opportunistic networks

DEPENG CHEN

Universitat Autònoma
de Barcelona

Anonymity and privacy on opportunistic networks

Author:
DEPENG CHEN

Advisors:
Dr. Guillermo Navarro Arribas
Dr. Joan Borrell Viader

Departament d'Enginyeria de la Informació i de les Comunicacions
UNIVERSITAT AUTÒNOMA DE BARCELONA

A dissertation submitted to the Universitat Autònoma de Barcelona in fulfilment of the requirements for the degree of Doctor of Philosophy in Computer Science.

SEPTEMBER 2020

Abstract

Opportunistic Networks are networks where communication happens opportunistically between nodes, end-to-end connectivity is not guaranteed and disruptions and delays are to be expected. There is plenty of work studying the routing strategies, architecture, dynamic congestion and network gestion, but not much work has been devoted to providing private and anonymous communication in these networks. However, it is pretty important to ensure user's privacy and provide anonymous communication in such networks as open access.

In this thesis, we have studied the anonymous and privacy technologies on opportunistic networks. More specifically, we divided the opportunistic networks into predictable pattern and generic pattern. To predictable opportunistic networks, we built an anonymous schema on onion routing. We designed two efficient routing finder algorithms. To generic opportunistic networks, we exploited mix networking strategy to ensure anonymous communication.

Furthermore, we have proposed several anonymous metrics to measure the anonymity property. We used the Jaccard distance to evaluate the routing path difference. Meanwhile, we exploited path degree measure to measure the anonymity degree of each routing path. To evaluate the performance of our proposed methods, we conducted all the experiments with real traces. From our observations, our designed methods can provide anonymous communication in opportunistic networks.

Resum

Les xarxes oportunistes són xarxes on les comunicacions es produeixen de manera oportunista entre nodes, no es garanteix la connectivitat de punt a punt i s'han de tolerar interrupcions i retards. Hi han molts estudis fets sobre estratègies d'encaminament, arquitectura, congestió dinàmica i gestió de xarxes, però no s'ha dedicat molta feina a proporcionar serveis de comunicació privats i anònims en aquestes xarxes. No obstant això, és força important assegurar la privadesa de l'usuari i proporcionar una comunicació anònima en aquestes xarxes. Com a problema afegit, aquestes xarxes solen utilitzar comunicacions sense fils, cosa que fa l'anàlisi del trànsit més fàcil que en xarxes cablejades més tradicionals.

En aquesta tesi, hem estudiat l'aplicació de tecnologies per l'anonimat i la privadesa en xarxes oportunistes. Més concretament, hem dividit les xarxes oportunistes en xarxes que presenten un comportament predictible i xarxes genèriques no predictibles. En el cas de les xarxes oportunistes predictibles, vam crear un esquema anònim basat en onion routing. En vam dissenyar dues estratègies d'encaminament eficients i segures per utilitzar onion routing. Per les xarxes oportunistes genèriques, hem utilitzat una estratègia basada en mix networks per garantir comunicació anònima.

A més, hem proposat diverses mètriques per mesurar l'anonimat. Aquestes mètriques mesuren diversos aspectes com la diferencia entre rutes d'encaminament de cara a la seva predictibilitat, o el grau d'anonimat d'una ruta concreta. Per avaluar el rendiment dels nostre mètodes proposats, hem realitzar tots els experiments amb dades basades en traces reals. A partir de les nostres observacions, els nostres mètodes dissenyats

poden proporcionar comunicacions anònimes en xarxes oportunistes.

ACKNOWLEDGEMENTS

At the end of my PHD research, I am very grateful to everyone who helped me during my studies. Firstly, I would like to thank for China Scholarship Council and UAB to offer me this grant to support my stay in Spain.

This research would not have been possible without my advisors Dr. Joan Borrell Viader and Dr. Guillermo Navarro-Arribas who were willing to give me tremendous support. I want to thank Dr. Joan Borrell Viader to offer me this opportunity to pursue my PHD degree in DEIC. He is a person who is always optimistic about work and life. He guided me how to enjoy the life and work smartly. Now he is suffering from health problems. I do hope he will soon get well again. I am also very grateful to Dr. Guillermo Navarro-Arribas to offer me the opportunity to work with him. He was very patient with me and spent a lot of time discussing research topics with me. He gave meticulous guidance with my research work. Also, he helped me a lot during my stay in Barcelona. I am really appreciate for my advisors' help.

I also want to thank Dr. Carlos Borrego Iglesias to help me starting a new research and give me lots of useful suggestions. I would like to thank Dr. Cristina Pérez Solà to help me improving my research work and discussed with me the potential future research. She and her family gave me numerous help to adapt to the local life. I want to express my gratitude to Roger and Victor, they offered me quite a lot of help when I was in trouble. Also, I had very good time with Carlos, Flora, Alicia and Enric. Thanks for sharing me the best local food.

To all teachers and colleagues in our department DEIC, I would like to express my thanks for all your help. I am very glad to work with you

and thanks for your valuable help. I also want to thank for all my Chinese friends during my stay in Barcelona. In those days with you guys, I would never feel lonely abroad.

Finally, I thank my family. During the four years I studied in Spain, they loved and encouraged me as always. Without their support, this thesis would be impossible.

Gracias por todos!

Gràcies per tot!

TABLE OF CONTENTS

List of Figures

List of Tables

CHAPTER 1

INTRODUCTION

With the development of network technology and the Internet, the connection between people and devices in the world is getting more prevalent and pervasive. Network technology in general is built upon some assumptions regarding the continuous connectivity and relatively static topology of the network and the devices that conform such networks. There are however environments and situations where we might find a dynamic network topology, or poor connectivity between devices. This means that the network will present long delays and even disruptions. The routing in such networks is somehow problematic due to this changing topology and potential delays and disruptions.

In these situations the use of typical network technologies such as the TCP/IP protocol stack is not convenient and might even not be possible. Unlike traditional communication networks, there is no stable connectivity in such extreme network conditions. Thus, the communication mode between each node is opportunistic. Some specific technologies have been developed to address this situations. Under the name of Opportunistic Networks (OppNets) or Delay-disruption Tolerant Networks (DTNs), these technologies allow to route messages and provide basic network services in the presence of disruptions and dynamic topologies. Applications for this networks include Terrestrial Mobile Networks [1], Airborne Networks [2], Military Ad-Hoc Networks [3], or Sensor/Actuator Networks [4].

In order to promote efficient communication in such opportunistic and delay tolerant

networks, most of the research has been focused on routing protocols [5–8], architecture [9–12], dynamic topology [13, 14], and network congestion [15–18]. However, not much work has been devoted to provide private or anonymous communications in these networks. The fact that these networks are usually wireless makes it relatively easy to monitor the traffic of the whole network and the possibility to provide some form of anonymity is undoubtedly interesting. Privacy is becoming an important issue in all kinds of communication networks nowadays and OppNets are also an interesting scenario, where the use of private communications can provide new applications and uses.

In this thesis we will focus on studying anonymous communications in OppNets. In general we consider a typical scenario in OppNets, where a node of the network sends a single short message to a destination node. We want to be able to hide the sender and/or destination addresses or identifiers from the message routing mechanism. In some sense this can be achieved in non opportunistic environments such as the Internet with techniques like onion routing and Mix networks.

We focus the work presented in this thesis precisely in the study of the application of onion routing and Mix networks to OppNets in order to provide anonymous communications in different OppNets scenarios. More precisely we consider generic OppNets and a the special case of predictable OppNets.

1.1 Generic opportunistic networks and predictable opportunistic networks

Opportunistic Networks (OppNets) are networks where communication happens opportunistically between nodes, end-to-end connectivity is not guaranteed and disruptions and delays are to be expected [19]. They are also denoted as Delay and Disruption Tolerant Networks (DTNs), although the term DTNs usually describes OppNets which use the specific Bundle protocol [20].

Among OppNets there are those where the contacts between the nodes follow an specific pattern [21]. In such cases the behavior of the network nodes can be predicted to some extend, and thus the communication and interactions between them can be known in advance. Such networks are usually denoted as Predictable Opportunistic Networks (POppNets). This predictability can be exploited to improve routing in the network for

instance. Predictability here, refers to the fact that the connectivity, the topology of the network, and its evolution over time, can be predicted ahead of time. We focus our work in the specific POppNet that raises from a network build on public transportation systems. Consider that all public buses in a city carry a simple network node allowing them to opportunistically exchange messages. Given the routes and timetables of the buses one can predict when interactions between nodes will occur during the day, and thus these networks can be used as a low cost urban networks. Routing can be more efficiently solved in POppNets than in generic OppNets due to their predictability, and we believe that some security services can also be improved. More precisely, anonymous routing is a difficult and complex problem in OppNets. Current solutions to provide anonymous routing in OppNets require complex cryptographic solutions, and complex setups. This is due to the fact that traditional solutions such as onion routing [22] cannot be directly applied in such networks. We will show however that the predictability in these networks can be exploited to actually use a simplified onion routing approach to provide anonymous routing for messages in POppNets.

1.1.1 Privacy and anonymity in OppNets

With the development of technology, more and more people are connecting on the network. In order to get more personal service, more and more applications online requiring our profile and other review caches. However, this somewhat leaks our information. Now, more and more people are caring about the personal privacy and want to communicate with others in an anonymous way. Privacy and anonymity have very similar meaning but with different objectives. From Wikipedia, privacy is the ability of an individual or group to seclude themselves or information about themselves, and thereby express themselves selectively. And Anonymity describes situations where the acting person's name is unknown. Some writers have argued that namelessness, though technically correct, does not capture what is more centrally at stake in contexts of anonymity. The important idea here is that a person be non-identifiable, unreachable, or untrackable. Anonymity is seen as a technique, or a way of realizing, a certain other values, such as privacy, or liberty.

1.2 Objective and contributions of the thesis

As a summary, the main objective of this thesis is to study the applicability of onion routing and Mix networks in OppNets. That is, the deployment of common techniques used in conventional networks to provide anonymity in the context of OppNets or DTNs.

One of the goals of the study was to be able to apply onion routing and Mix networks solutions in a direct and simple manner. We aimed to provide solutions without the need for complex cryptographic schemes or use approaches designed ad-hoc for very specific scenarios. To that end the use of onion routing and Mix networks is relatively simplified, focusing the work on the evaluation and the achieved privacy and specific nature of the networks such as their dynamic behavior.

This has been articulated through two main objectives, and therefore results, which are the outcome of the work performed during the thesis.

1. Onion routing for POppNets. Given the predictability on network behavior and the possibility of using, to some extend, source routing, we will show that the use of onion routing is not only possible but can also be a good solution for anonymous communications.
2. Mix networks for general OppNets. In the case where we lack the avility to perform some sort of source routing we will show that a good alternative is the use a solution based on Mix networks.

In all cases we have considered and evaluated the proposals using realistic data. For the case of POppNets we use a network created from the public transportation buses of a big city, Seattle. Buses carry a network node and contacts between nodes are produced when two buses are in range. The fact that node, thus buses, follow a fix schedule makes the network behavior predictable to some extend. There will obviously be errors in the prediction due to traffic issues but in general is considered a good example of POppNets. For the evaluation of generic OppNets we have used different datasets obtained from real network traces which are commonly used in OppNet research: mobility traces from people obtained during a conference, contacts from students of a research group within a university campus, and the contacts from taxis in a big city. With this approach we try to validate our results with common scenarios commonly used by the research community. There will be obviously very specific cases of OppNets and POppNets which differ from

those we have used, but in general the datasets are considered relatively standard in the community.

1.2.1 Publications

As a result of the thesis we have produced the publications listed here.

- **D. Chen**, G. Navarro-Arribas and J. Borrell: On the Applicability of Onion Routing on Predictable Delay-Tolerant Networks, 2017 IEEE 42^{nd} Conference on Local Computer Networks (LCN), Singapore, 2017.

  In this paper, we proposed the idea of using an onion routing based method to provide anonymous communication in predictable delay tolerant networks. We introduced a random algorithm to find the routing path and our results showed the good performance of using such strategy. This is the start of our work and gave us essential support for our future work.

- **D. Chen**, G. Navarro-Arribas, C. Perez-Sola and J. Borrell: Message anonymity on predictable opportunistic networks. Journal of Ambient Intelligence and Humanized Computing, 2019.

  In this paper, we presented and formalized the main contribution related to the use of onion routing in POppNets. We introduced a total random path finder and forward-backward finder routing strategies to ensure the anonymous communication in predictable OppNets. We analyzed the anonymity performance and the efficiency of these methods with experimental results. This work will be mainly presented in chapter 3.

- **D. Chen**, G. Navarro-Arribas, C. Perez-Sola and J. Borrell: A review of Message Anonymity on Predictable Opportunistic Networks. In Actas de las V Jornadas Nacionales de Investigacion en Ciberseguridad (JNIC2019), pages 316-317, C aceres, Spain, June 2019. Universidad de Extremadura.Servicio de Publicaciones

  This paper is an extended summary of the previous journal publication.

- **D. Chen**, C. Borrego and G. Navarro-Arribas: A privacy-preserving routing protocol using Mix networks in Opportunistic networks, *under review, 2020*.

In this paper, we introduced the Mix networking based method to ensure anonymous communication in generic opportunistic networks. This work will mainly be described in chapter 4.

1.3 The organization of this thesis

There are five chapters in this thesis. The structure of this thesis is organised as follows. This Chapter 1 is a general introduction of this work's motivation and contributions. This chapter gives the background of the whole thesis and points out the importance of our work. In chapter 2, we introduce preliminaries techniques with some details and some related work on this thesis. In chapter 3, we focus on onion-based mechanism to ensure anonymous communication in predictable OppNets, and, in chapter 4, the Mix networking mechanism is used for generic OppNets. Chapter 5 gives a conclusion of this thesis and points out the potential future work.

CHAPTER 2

PRELIMINARIES AND RELATED WORKS

The preliminaries and related work will be introduced in this chapter. As previously stated, the main objective of this thesis is to enhance privacy and anonymous communications in Opportunistic networks (OppNets). As we will show, OppNets can be roughly divided into predictable OppNets and general OppNets. Section 2.1 will give an introduction of general OppNets, and in Section 2.1.1, we will introduce predictable OppNets. We will provide an overview of routing strategies in OppNets in Section 2.1.2.

The work of the thesis, that will be presented in the following chapters, will introduce some solutions for anonymous communications in both predictable OppNets and general OppNets. There are two main anonymous mechanisms used in this thesis: onion routing-based and Mix networking-based methods. More precisely, we will adapt them to its use in different OppNet scenarios. Section 2.2 will give an overview of these two anonymous communication mechanisms. Section 2.3 summarizes related work on privacy and anonymity for OppNets.

In our contributions we will use a formalization of OppNets as dynamic graphs, which we will introduce in Section 2.4.

2.1 Opportunistic Networks

In Opportunistic Networks [23] we usually assume that users carry mobile devices to communicate with each other and data is exchanged between those devices. The communications are performed in an opportunistic manner, and high delays and disruption are to be expected.

We can differentiate between two different types of OppNets depending on the predictability of their topology and behavior: predictable and non predictable OppNets.

A generic OppNet, or non-predictable OppNet, is a network where the contacts between nodes cannot be predicted. This is the general case, usually considered when one deals with OppNets of DTNs. We have considered think type of networks in Chapter 4, but we also considered the special case of predictable OppNets.

2.1.1 Predictable Opportunistic Networks

Among OppNets, there are those where the contacts between the nodes follow an specific pattern [21]. In such cases the behavior of the network nodes can be predicted to some extend, and thus the communication and interactions between them can be known in advance. Such networks are usually denoted as Predictable Opportunistic Networks (POppNet). This predictability can be exploited to improve routing in the network for instance. Predictability here, refers to the fact that the connectivity, the topology of the network, and its evolution over time, can be predicted ahead of time.

The study of POppNet in the literature is usually focused on improving routing as compared to generic OppNet. POppNet appear in specific scenarios, such as satellite networks, public bus networks, or even human mobility [24]. In this sense, one of our main research work is in the specific POppNet that raises from a network build on public transportation systems (see Chapter 3). For example, consider that all public buses in a city carry a simple network node allowing them to opportunistically exchange messages. Given the routes and timetables of the buses one can predict to some extent when interactions between nodes will occur during the day. These networks can be used as a low cost urban networks. or even emergency backup networks in case of failure of traditional infrastructure networks. Routing can be more efficiently solved in POppNet than in generic OppNet due to their predictability, and we believe that some security services can also be improved. More precisely, anonymous routing is a difficult

and complex problem in OppNets. Current solutions to provide anonymous routing in OppNets require complex cryptographic solutions, and complex setups. This is due to the fact that traditional solutions such as onion routing [25] cannot be directly applied in such networks. We will show however that the predictability in these networks can be exploited to actually use a simplified onion routing approach to provide anonymous routing for messages in POppNet in Chapter 3.

2.1.2 Overview of Routing protocols in OppNet

Among all proposed routing protocols for OppNets, there are three main categories: message-ferry-based, opportunity-based and prediction-based [26]. We summarize here the main routing strategies in order to take a quick look on how routing works in OppNets as opposed to common computer networks. The particularities of such routing problems is what lies in our proposals to provide anonymous communications for OppNets. In our case, as we will see, we will use simple routing protocols in our proposals in order to focus our efforts in the privacy part of the routing itself.

2.1.2.1 Message ferry based routing protocol

Message ferry based routing protocols make use of an extra node, referred as ferry, to assist in message forwarding [27]. In this schema, the ferry moves around its deployed area with a known trajectory, and its responsibility is to carry the message received from regular nodes and forward it to destination nodes or other ferries. In such kind of ferry based schema, the control of the route and mobility of the ferry has a significant impact on the performance of the whole system [28].

There are two different approaches in these routing protocols: single ferry and multiple ferries. The advantage of using a single ferry is that the route design is easy to control. However, it is more vulnerable to ferry failures. With multiple ferries system, it would be more complex to control the routes and it will consume more energy.

The ability to deploy a ferry will depend on the specific network and scenario. In the general case it might be difficult to assume that we will have a ferry or the possibility of using one.

2.1.2.2 Opportunistic-based routing protocol

Opportunistic-based methods use every node in the system to forward the message from one node to another until reaching the destination [29, 30]. Usually, when two nodes are in their communication range, they can exchange messages. There are roughly two different ways to deliver messages: one copy and multiple copies. In one copy mode, each node will forward only one copy per message to the system. One copy mode can significantly reduce the resources but it leads to low delivery ratios and high latencies. Another way to delivery messages is to exploit multi-copy mode which forwards multiple copies per message. The multi-copy mode increases the message delivery ration but can take up a lot of network resources. Thus, one should choose between this two modes according to its network properties and requirement.

One of the most popular routing protocols for OppNets i the so called *epidemic routing*, which is a type of multi-copy opportunity routing. In this case a node that receives a message and has to forward it, will always perform a broacast to all contacted nodes during some limited period of time. This type of protocols usually provide relatively good delivery ratios and latencies at the cost of generating traffic in the network.

2.1.2.3 Prediction based routing protocol

The above two types of protocols assume that the mobility of nodes in an OppNet is totally random. However, in some specific scenarios, the movement pattern is predictable. We have seen POppNets where we can predict the whole network behavior, there are alos hybrid cases where we can predict only some aspects of the topology and network behavior over time.

In the first case, we can use time based source routing. The source node can predict the future evolution of the network and find a path that will deliver its message to the destination. There is always a failure rate to be expected due to possible errors in the predictions, but in general routing it can be relatively efficient.

In other cases, where the full network behavior cannot be predicted, there is usually some information that can be used to model the future behavior of nodes or contacts. For example, if a node has visited an specific place in the past, it will more likely visit it in the future. In [31], Juang *et. al* proposed a history-based protocol to relay the data in OppNets, where the basic assumption is that nodes that were previously within range of

the base station will still be close in the future.

Lindgren *et. al* proposed a probabilistic routing strategy called PROPHET which uses a delivery probability metric to compute the probability of two nodes encountering [32]. PROPHET has three different situations to update the delivery predictability: the encountering of two nodes, the time (or age) the two nodes having not encountered and the intermediate frequency encountering node. The main idea of updating the delivery predictability is that the more frequent two nodes meet, the higher the delivery predictability between them will be. In PROPHET, a message is transmitted from one node to the other if the delivery predictability to the destination node is higher than a given threshold.

MaxProp [33] uses the rank of packets to decide whether to transmit the packet or delete it. In order to rank its packets, a node in MaxProp firstly divides all packets received in its buffer storage into two categories according to the number of hops they have traversed. If the count for a given packet is less than a specific threshold, the packet belongs to the high rank set, which will be transmitted first in a short hop count order. Otherwise, packets will belong to the low rank which will more likely be deleted.

Liu and Wu presented a hierarchical routing approach to ensure the scalability and delivery of OppNet[34]. In their schema, they assume that the contacts in the network are predictable. To ensure the scalability and delivery ratio, they use multilevel clustering and hierarchical routing. The multilevel clustering and hierarchical characteristics help the network build its level topology which is composed of cluster heads on different levels. The main idea of this routing method is to find the common cluster head on the higher level. And then it traverses the lower level to find the next hop to forward the message.

2.1.2.4 Other proposed routing protocols

In [35], Chen *et. al* proposed a coding and replication based routing method. Coding based routing strategy divides the message into several small blocks, and once the destination node received all the small parts, the original message can be reconstructed by these small blocks.

2.2 Anonymous Communication

There are a variety of technologies trying provide anonymous communication. Onion routing and Mix networks are two of the most popular solutions in communication networks. Onion routing uses layered encryption to wrap the message to hide the transmitting flow. As the message is transmitted, each node on the routing path should peel of (or decrypt) it in a sequential order. A Mix network, on the other way, uses Mix nodes to collect a set of messages and shuffle them before sending them to the next hop. Each Mix node will wait for a threshold number of messages and forward them at the same time. So, the messages in a Mix network do not need to be synchronous.

These two techniques are commonly deployed in computer networks where connectivity is guaranteed but might present some problems when considering an OppNet scenario. As we will see in this thesis several adaptations are proposed in order to consider the use of onion routing and Mix networks in OppNets.

2.2.1 Onion routing

Onion routing [36] is commonly used to provide anonymous communications for computer networks such as the Internet. An onion routing schema is composed of a set of Onion Routers (ORs). Each user chooses several ORs in order to construct an anonymous channel or *circuit*. The main idea of onion routing is to use layered encryption to encrypt the original message so each OR on the path peels off its layer and forwards the message to the next OR or destination.

The message, encrypted with different layered keys, ensures that each router on the path only has knowledge of its previous and next hops. Thus, only the entry OR has the knowledge of the message origin or user identity, which has constructed the forwarding message. It is important to note that by user identity we refer to the network address of the user. That is, we are only protecting the origin of the message. Whether the message content discloses information about the user or origin is out of the scope of these frameworks.

Figure 2.1 shows a simplified example of message transmission using onion routing. Here, the sender wants to send message MSG to the receiver. The routing path is composed of 3 onion routers. So, firstly the sender will establish the virtual circuit by using these three onion routers' public keys to encrypt the message and then forward to

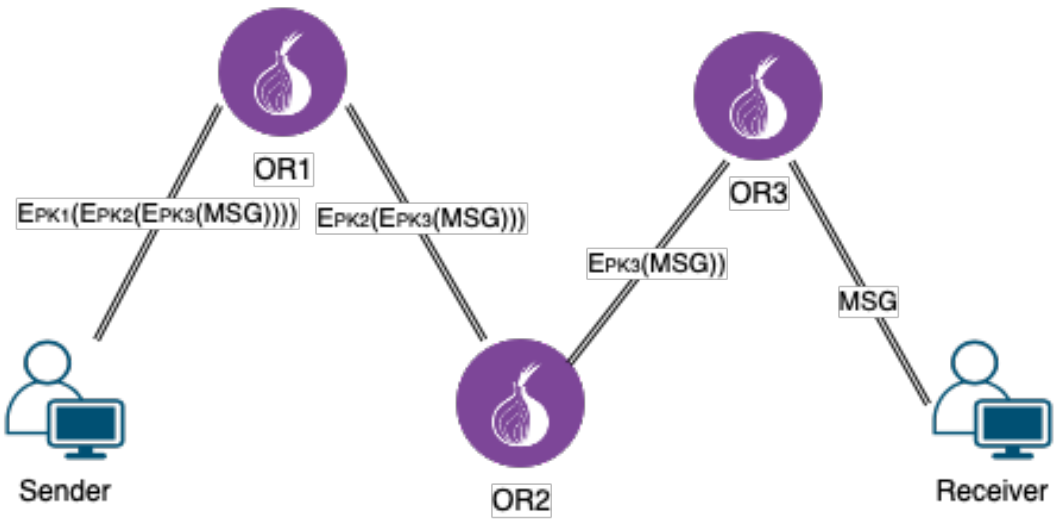


Figure 2.1: An example of message transmission in onion routing

the next hop. Each OR has its own private and public key. The sender first constructs a layered encryption message with these three OR's public key PK_1, PK_2, PK_3. Thus, the encrypted message transmitted to $OR1$ is $E_{PK_1}(E_{PK_2}(E_{PK_3}(MSG)))$. Router 1 receives the triple encrypted message and uses its private key $SK_1$to decrypt and peel off the first layer onion message. Router 1 parses the data package and gets the forwarding destination router 2 and the message $E_{PK_2}(E_{PK_3}(MSG))$. And then it forwards the 2 layered encrypted message to router 2. The following routers are doing the same procedure. And the receiver gets the message MSG from the last onion router $OR3$. In this process, the intermediate node can only know its preceding and following nodes and cannot establish the link between the sender and the receiver because of the multi-layered encryption(usually at least 3 layers).

2.2.1.1 Message forwarding

There are mainly three phases in onion routing, i.e., connection initiation, data forwarding, and connection termination. In the first phase the source node stablishes a path through at least three intermediary onion routers (ORs). It collects the public keys of the ORs in the path to set up the $circuit$. And then the source node sends the message through the $circuit$. Each OR receives the message and peels off its layer with its private key. Before sending the message to the next hop, each OR it will pad the message to maintain a fixed size. The communication process can be terminated by any

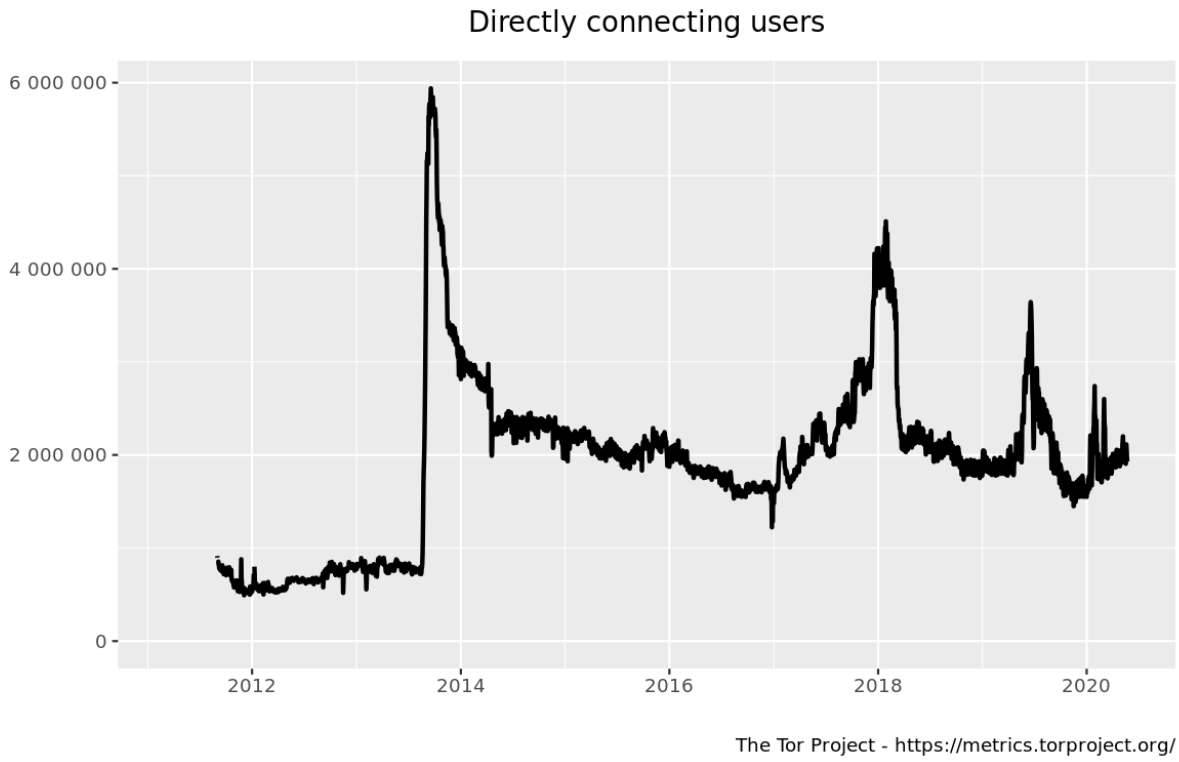


Figure 2.2: The estimated number of directly-connected users around the world

OR on the routing path or the final destination node. To prevent an eavesdropper or any compromised OR to trace the message flow, all messages should use a uniform size, making it hard to identify each message according to its size [37].

One of the core issues before message forwarding is the constructing the layered encrypted message. Anonymity is ensured because each OR in the routing path has limited knowledge of the path. There are three kinds of routers in a path, i.e., entry, middle and exit. The middle OR only has the knowledge of the other two OR neighboring nodes. Also the $entry$ and $exit$ routers have partial information of the message flow. Since the size of each message is fixed, it is impossible to identify the two parties communication according to its message structure.

As an example of the use of onion routing we find the Tor project [38]. Tor uses onion routing to provide anonymous communications in the Internet, by defining the protocols and providing an open network of ORs. The popularity of Tor can be seen in Figure 2.2.

As in common onion routing schemes, there are three different ORs composing a Tor $circuit$ of three ORs: entry guard OR, middle OR and the exit OR. The entry guard OR

is the first router connecting to sender and the exit OR is the router closer to the receiver. The middle ORs are other routers on the routing path between the entry guard and the exit node. In order to improve the security and performance in the Tor network, there are plenty of works focused on path selection mechanism [39–41]. Before sending the message in the Tor network, the user should choose the onion routing path first. The path is chosen based on several criteria such as node bandwidth, network congestion, and availability. It is important however to maintain a certain degree of randomness in the selection of nodes to increase anonymity [39].

Choosing random onion routers can ensure higher anonymous degree. However, this not the case in some scenarios. In [42], Roger **et. al** pointed out that using a single fast entry OR is better than three different entry guards, which means only choosing random entry guard for each connection can not ensure the best anonymous. The fast entry guard, which can ensure enough bandwidth to forward message and provide high degree of anonymity. This means that routers with different capacity can largely affect the performance and anonymity of the Tor network. Only randomly selecting OR will not be good enough for the anonymous communication. This is because that reducing the number of guard nodes can reduce the probability that the nodes are compromised by the relay adversary.

As from the above introduction of the Tor network, we can see that there is some difficulty on balancing security and performance. Other than improving the performance and security of Tor network, there are some works investigating the weakness of the system. Because of the limited number of Tor Onion services and resources, it is particularly vulnerable under fingerprinting attacks. Rebekah **et. al** analyze the impact of fingerprinting attacks on Tor Onion services[43]. Fingerprinting attacks exploit the traffic traces to link the sender and receiver. Adversaries collect the fingerprint of target website or service and compare it with the with the victims' traffic. Since the openness of the network service, adversaries can observe the victim's traffic with economic resources.

2.2.2 Mix Networks

Chaum [44] first proposed the concept of Mix and Mix network in 1981. A Mix network (MixNet) is an anonymous communication protocol where each proxy node (referred as Mix) shuffles a bunch of messages from multiple senders before forwarding them to the next destination [44]. To ensure the shuffling, each message will be encrypted with

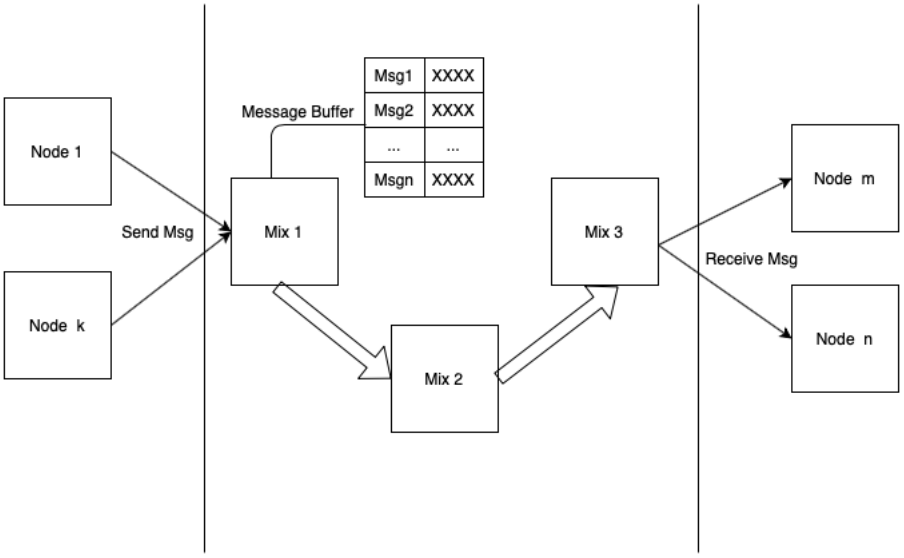


Figure 2.3: An example of message transmission in Mix networking

the public key of the Mix servers. Similarly to onion routing, Mix servers decrypt each arriving message with its private key.

There are different approaches to implement a Mix network but the main structure is quite similar in all cases. Figure 2.3 broadly shows how a Mix networking works.

Before sending the message to the Mix network, the sender will create a onion-like encrypted data structure with each Mix node's public key. upon receiving the message, each Mix node will peel off its current layer with private key and forward it to the next Mix node. This operation is repeated until the last Mix node extracts its destination address. The last Mix node decrypts the message and transfers it to the destination user. The message flow is very similar to the onion routing process. The difference is that each Mix node waits until it has received a given number of messages before forwarding them to the next node. The forwarding of all messages is performed at the same time giving the effect of shuffling the messages. That is, making it hard for an external observer to correlate input and output messages.

There are two main related parameters or factors in a Mix network, i.e., the number of messages it collects and the delay it needs to send these messages [45]. If the node receives a high load of messages the delay will be shorter, while if it receives few messages will produce higher delays. These two factors impose a trade-off between performance

and anonymity.

2.3 Privacy for Opportunistic Networks

Anonymous communications have been studied in the context of Opportunistic networks and Delay-tolerant networks. In general, due to the lack of end-to-end connectivity on these kind of networks, security solutions for them are difficult and complex. Furthermore, it is commonly assumed that one cannot relay on common solutions from more conventional connected networks due to the need to support disruptions and delays.

In this regard, [46] proposes a security architecture for DTNs using Identity-Based Encryption (IBE) with support for anonymous authentication with pseudonyms. Anonymous routing is somehow achieved by using gateways that hide the sender/receiver.

A common approach in several works is to apply an onion routing strategy in DTNs by creating onion groups. That is, nodes are grouped and onion layers are based on those groups. Any node of the group can forward the message of the corresponding layer. This usually implies that all nodes of the group can decrypt the layer. ARDEN [47] uses groups of nodes to perform the layering process and broadcast messages between these groups. The message route between groups is chosen randomly. To perform the cryptographic layers attribute based encryption (ABE) is used. Similarly, [48] also uses onion groups and allows several copies of the message. In [49], authors allow such groups to be dynamic, but use straightforward public key cryptography, having all members of the group sharing the same private key. Another interesting work is [50]. Here authors want to hide the physical location of the sender. They propose to fragment the message and send the fragments to different receivers, with the aim of creating confusion for the attacker.

In this line, another approach to achieve anonymous communications in dynamic networks in general, is precisely to introduce noise in the communications. This approach is very common in traditional data privacy (including Statistical Disclosure Control, or Privacy-preserving Data Mining) [51]. As an example [52] attempts to ensure differential privacy for several observable characteristics or metadata. These approaches are quite different from the ones presented in this thesis, and can be seen as alternate methods. We believe that knowing the network behavior in advance makes our approach more suitable for fast transmission of single short messages.

In [53], Rui *et. al* present a protocol based on social relationship and encryption to provide location privacy in OppNet. This work also presents the problem where, in order to save energy, storage and computing resources, some nodes may behave selfishly and might not be willing to participate in relaying. In IPAD [54], each node exploits a family of unlinkable pseudo-IDs and periodically changes its current pseudo-ID for privacy preservation.

In [55], authors proposed a multiple encryption method to provide privacy content based opportunistic networks. The main idea is to use an onion routing-liked multiple layer commutative encryption to encrypt the message several times with different keys. Hence, the intermediate nodes can only decrypt one layer without leaking the metadata of the message.

Wang *et. al* uses attribute-based encryption and a security-based mobility prediction method to provide privacy and improve the forwarding efficiency [56]. A $k-$order Markov chain of encountered nodes can be trained to predict the next contact. The node with more relevant centrality will be chosen as the next forwarding node.

In [57], Zakhary *et. al* use a Markov-based stochastic model for location prediction, and use a k-anonymity-based protocol to ensure better privacy. Using the Markov model to predict the routing path. Due to the lack of stable connections, the message forwarding process needs to exploit any possible encounter to transfer it. Meanwhile, the OppNet mobility is usually modeled as a random walk. Therefore, the Markov-based stochastic model can be chosen to predict such kind of behavior.

2.3.0.1 Anonymous communication in predictable opportunistic networks

In the special case of POppNets, the predictability of the network has been used to improve routing performance [58–60]. The use of such predictability to improve or design security related services or mechanisms in the context of OppNet has not be exploited yet to our knowledge.

Most of the existing works on anonymous communications deal with general OppNet, where predictability of network behavior is not considered. We are not aware at the time of writing about publications on anonymous communication in POppNet. Solutions from non predictable networks can obviously be applied to POppNet, but taking into account its predictability allows to simplify and improve the anonymous routing in such

networks.

2.4 A model for Opportunistic Networks

In this section we will introduce a model for OppNets based on dynamic graphs. This model will be used in the following chapter to present our contributions. The model allows to abstract the main parameters and particularities of OppNets and POppNets and helps in formulating the problems that we will present.

An OppNet can thus be modeled as a dynamic graph. That is, a graph with dynamic components, which in our case are the presence of edges and vertices or nodes. Vertices represent network nodes and edges represent the fact that there is a connection between two nodes. These graphs are also denoted in the literature as temporal networks [61] [62], time-varying graphs [63], temporal graphs [64], or evolving graphs [65]. In our case, each edge has a temporal presence based on the time that a connection can be established between the two nodes, usually based on the coverage of the network nodes.

For our scenario we consider the dynamic graph to be undirected because a connection between two nodes represents the fact that these two nodes can communicate in both directions. We denote such an undirected dynamic graph as $G(V,E)$, where V is the set of nodes, and E is the set of edges. Each edge, denoted as $e = (u,v,t,\lambda)$, is a *temporal edge* between nodes u and v, starting at time t, with a duration of λ. Note that $(u,v,t,\lambda) = (v,u,t,\lambda)$. For simplicity we will use $\lambda(e)$ as the duration of the link in edge e, and $t(e)$ as the starting time for edge e, or simply λ and t if the edge can be easily inferred from the context. We will not consider the timed presence of nodes. Note that a node presence can be modeled by eliminating all its edges during the time when the node disappears.

We also consider a unique transmission time τ for all messages to be sent in the network. This can easily be extended to a variable transmission time specific for each message, or edge. The transmission time includes the time required to establish the connection, send, and receive the message. We also assume that for any edge $e \in E$ for a given graph $G(V,E)$, $\lambda(e) \geq \tau$. That is, all edges in the graph can be used to send a message. Edges that do not address this constraint can be removed from the graph in a pre-processing step.

The graph $G^*(V^*,E^*)$ is the static undirected graph obtained from G by considering

all its edges without time constraints (all edges are present in the graph independently of time). We use $N(v)$ to denote all neighbors of node v in G^*. That is, $N(v) = \{u \mid (v,u,t_i,\lambda_i) \in E^*, \forall i\}$.

We provide some definitions here that will be used in the thesis.

Definition 1. *The* time forward neighbors *or future neighbors of a given node $u \in V$ at time t_c for the dynamic graph $G(V,E)$ are defined as: $N^{t+}(u,t_c) = \{v \mid (u,v,t,\lambda) \in E, t_c \leq t+\lambda-\tau\}$.*

Similarly,

Definition 2. *The* time backward neighbors *or past neighbors of $v \in V$ at time t_c are defined as: $N^{t-}(v,t_c) = \{u \mid (u,v,t,\lambda) \in E, t+\tau \leq t_c\}$.*

In general, $N^{t+}(u,t_i) \cup N^{t-}(u,t_i) = N(u)$. From a practical point of view, we usually want to consider time forward and backward neighbors up to a maximum delay or carry time. If a node wants to deliver a message to another node, the first one can carry the message up to a maximum time. In such case, for simplicity we define a unique maximum carry time for all nodes denoted as δ_M. The extension to a variable carry time depending on the node is straightforward.

Definition 3. *The* time forward neighbors with carry *of node $u \in V$ at time t_c considering a maximum delay or carry time δ_M for the dynamic graph $G(V,E)$ are the nodes: $N^{t+}_{\mathscr{C}}(u,t_c,\delta_M) = \{v \mid (u,v,t,\lambda) \in E, t-\delta_M \leq t_c \leq t+\lambda-\tau\}$.*

Definition 4. *The* time backward neighbors with carry *of node $u \in V$ at time t_c considering a maximum delay or carry time δ_M for the dynamic graph $G(V,E)$ are the nodes: $N^{t-}_{\mathscr{C}}(u,t_c,\delta_M) = \{v \mid (u,v,t,\lambda) \in E, t+\tau \leq t_c \leq t+\lambda+\tau+\delta_M\}$.*

We also consider the degree of a node at a given time. That is, the degree of a node will be time dependent. In our case we are interested in looking for the degree associated to time forward neighbors with carry, and time backward neighbors with carry. We thus define the time forward degree and time backward degree of a node as follows.

Definition 5. *The* time forward degree *of a node u at time t_c with a maximum delay δ_M is defined as: $deg^{t+}_{\mathscr{C}}(u,t_c,\delta_M) = \mid N^{t+}_{\mathscr{C}}(u,t_c,\delta_M) \mid$, and the* time backward degree *is defined as $deg^{t-}_{\mathscr{C}}(u,t_c,\delta_M) = \mid N^{t-}_{\mathscr{C}}(u,t_c,\delta_M) \mid$*

$P = \langle (v_1,t_1),(v_2,t_2),\ldots,(v_l,t_l)\rangle$ denotes a *path* in the dynamic graph $G(V,E)$, where (v_i,t_i) represents the node in the path and the time that the message arrives to such node. Given the path P the *length* of the path is the number of nodes included in path. The *duration* of a path P is the time taken by the message to arrive to destination. That is, for $P = \langle (v_1,t_1),\ldots,(v_l,t_l)\rangle$, $duration(P) = t_l - t_1$.

2.5 Conclusions

In this chapter we have briefly introduced the base and building blocks of our proposals. We have described OppNets and the particular case of POppNets including an overview of common routing strategies. As we will see in the following chapter our work is within the application of onion routing and Mix networks to these kind of networks. We have also seen that although there exists related work on anonymous communications for OppNets, it usually concerns complex setups and cryptographic mechanisms. We will see that our approach is to make use of simplified onion routing and Mix networks. This simplification has the drawback that the scenario of application is restricted and will introduce some performance penalty in message routing. As we will demonstrate, the penalty is however tolerable in most situations.

Finally we have introduced a model for OppNets based on dynamic graphs, which serves as a generic abstraction over which we will present our contributions in the following chapters.

CHAPTER 3

ANONYMOUS COMMUNICATIONS IN POPPNETS

When considering privacy preserving routing, both OppNets and POppNets present some challenges. Given its nature it is clear that common privacy-preserving routing strategies such as Onion Routing are difficult to implement directly. In this chapter we consider privacy preserving routing in POppNets.

In the case of POppNets, the fact that we can predict the future network behavior can enable the use of source routing. The sender can select the path that the message will follow to reach its destination, which could enable the use of Onion Routing. This is of course a bad option for onion routing, since an attacker could use the known network topology to infer information about such path. We need thus, to design routing strategies for POppNets, which can introduce enough randomness to be hard to guess by an attacker. In such case, we argue that Onion Routing could be a solution to anonymous routing in POppNets.

In this chapter, we will present two stochastic routing strategies which could be used to implement Onion Routing in POppNets. We will also introduce some anonymity metrics, and provide an evaluation of the performance of the proposed solutions using data from a public transportation network.

3.1 Predictable OppNets

Routing in OppNets is a difficult problem due to the unforeseeable nature of contacts between nodes, which leads most of the times to the adoption of (limited) broadcast approaches for routing [66–68]. However, the fact that we are dealing with predictable opportunistic networks (POppNets) greatly simplifies such routing. Routing can be seen as establishing paths in the time-based dynamic graph that represents the network [21]. Solutions exists to determine shortest paths in such networks. This is however not a good solution from a privacy perspective if we consider an onion routing strategy. In this case, the predictability of the chosen path is a key issue that needs to be protected. We introduce two stochastic strategies to determine paths in dynamic graphs, and thus, POppNets. The main objective of these approaches is to determine a path to be used for onion routing in such networks. The stochastic nature of both algorithms has the goal of keeping the predictability of the path as low as possible.

3.2 Privacy-preserving routing strategies in POppNets

In this section we introduce our proposed strategies to establish a source path with enough randomness to be used in onion routing. The first one is based on performing a constrained time-based random walk. In the second case we attempt to improve the path efficiency in terms of length and delivery time by providing a more efficient search of possible paths.

In order to evaluate our proposals we can compare the use of the selected paths to the optimal solution. That is, the path that we will select if we where not going to use onion routing. The common approach is to look for the shortest path from source to destination. We will also present this routing strategy in the following section. This path is obviously a bad choice to implement onion routing since it can be predicted by an attacker having knowledge of the network behavior. Instead it will be used as a benchmark to evaluate our proposed algorithms.

3.2.1 Optimal routing

There are several options to find an optimal path in a dynamic graph. Such optimal path can be determined based on different criteria. We can look to optimize the path

length (shortest number of hops), the shortest time path (or foremost), or the fastest path (earliest arrival). We have used the shortest path optimal algorithm as it provides a good simple base.

The shortest path in a dynamic graph can be seen as an adaptation of the well known Dijkstra shortest path algorithm to the case of dynamic graphs. That is, taking into account the time based evolution of the graph. To that end, the algorithm exploits the fact that given a shortest path from s to d arriving at time t, then the prefix path from s to u, where u is the node in the path just before d, is shorter than all paths from s to u ending before t [65]. That is, given a shortest path $P = \langle (s,t_0),\ldots(u,t_u),(d,t_d)\rangle$ then the prefix $\langle (s,t_0),\ldots(u,t_u)\rangle$ is shorter for $t_u < t_d$.

Data: $G(V,E)$ is a dynamic graph; $s \in V$ is the source node; $d \in V$ is the destination node; t_s is the starting time; T is the tree of pairs (v,t)
Result: The shortest path tree T.
begin
  $T = (s,t_s)$;
  $earliest(s) = 0$;
  $earliest(u) = Max$;
  $d = 0$;
  $location(s) = (s,t_s)$;
  while $location(u) == NULL$ *and* $u \in V$ **do**
    foreach *(u,t) in tree at depth d* **do**
      Get all the neighbour pairs (v,t_c) of (u,t);
      if *location(v)==NULL* **then**
        location(v) = (v,t_c)
      end
    end
    if *earliest(v)* > t_c **then**
      earliest(v) = t_c;
      (v,t_c) is a child of (u,t) in T;
    end
    d = d + 1;
  end
  return T;
end

Algorithm 1: Optimal routing path finder.

The actual algorithm in shown in Algorithm 1. Here, the $earliest(v)$ is the function to find the earliest path of node v, which tells when node v is arrived earliest in the routing path tree. $location(v)$ is the function to find the location of node v. Since a node v might appear several times in routing path tree, $location(v)$ will tell where node v in the tree to represent the shortest journal from source node s to node v.

3.2.2 Random Path Finding

The first strategy we propose to use in order to apply onion routing in POppNets is to use a time-based random walk on the graph. We denote it as *random path finding* and it is intended to ensure that the predictability of the path is very low. The idea is that this will be somehow similar to randomly choosing the onion routing nodes in Tor.

Data: $G(V,E)$ is a dynamic graph; $s \in V$ is the source node; $d \in V$ is the destination node; t_s is the starting time; t_c is the current time; L_{min} is the minimum path length; L_{max} is the maximum path length; τ is the transmission time;
Result: Path P from s to d, such that $L_{min} \leq length(P) \leq L_{max}$.
begin
  $P = \langle (s, t_s) \rangle$;
  $t_c = t_s$;
  $continue = True$;
  while $continue$ **do**
    u = getLastNode(P);
    select a random edge $e = (u, v, t, \lambda)$ such that $e \in N^{t+}(u, t_c)$;
    if $t_c > t(e)$ **then**
      $t_c = t_c + \tau$;
    end
    else
      $t_c = t(e) + \tau$;
    end
    $P = P + \langle (v, t_c) \rangle$;
    if v *is repeated in* P **then**
      P = mergeNodePath(P, v);
    end
    if $v = d$ *and* $L_{min} < length(P) < L_{max}$ **then**
      $continue = False$;
    end
    else if $N^{t+}(u, t_c) = \emptyset$ *or* $length(P) > L_{max}$ **then**
      $P = \langle (s, t_s) \rangle$;
      $t_c = t_s$;
    end
  end
  return P;
end

Algorithm 2: Random path finder.

The algorithm, shown in Algorithm 2, allows inclusion of repeated nodes but performs a shrink or merge of the path if a repeated node is found (`mergeNodePath(`P`,`v`)`), eliminating the intermediate nodes. This intermediate nodes do not provide additional security in the path given that the repeated node could correlate traffic received twice.

For example, in path $P = \langle (v_0, t_0), (v_1, t_1), (v_2, t_2), (v_3, t_3), (v_4, t_4), (v_5, t_5) \rangle$, if we know that node v_2 is the same as node v_4, the resulting merged path will be: $P = \langle (v_0, t_0), (v_1, t_1), (v_2, t_2), (v_5, t$

That is, node v_2 carries the message until time t_4 to send it then to node v_5. We have found this approach to be faster than just backtracking or re-starting the path search if a repeated node appears.

For practical reasons we bound the search to prevent very long paths (both in time and number of nodes). The search is restarted in no path is found within the constraints. This bound can be set based on different parameters: maximum path length, duration, computation time, ... In our case, we have set the bound to a maximum path length (L_{max}) and a maximum number of executions of the main loop (set to $10M$ in the experiments of the evaluation). As we will see in Section 3.4 the algorithm produces paths with reasonable length and duration.

3.2.3 Forward and Backward Path Finding

In order to improve the random path finding algorithm, we have designed another stochastic approach using a meet-in-the-middle strategy. The idea is to use information from both source and destination nodes to perform a stochastic path search from both nodes at the same time. As we will see, this approach is more likely to find the path in case it exists and can produce shorter paths.

This algorithm is denoted as the Forward-Backward path finder (FB), and is shown in Algorithm 3.

The algorithm performs a partial path search starting from the source node, where a path will be randomly selected from potential forward neighbors. At the same time the analogous procedure is performed from the destination node selecting backward neighbors. Actually, to increase uncertainty, the algorithm keeps track of all potential forward and backward partial paths to finally randomly select an intersection, yielding the final path. In order to speed up the process, both forward and backward searches are done at the same time, so each step of the algorithm increases the path length by two new nodes.

In Algorithm 3, the partial paths are kept in an array-like structure, $P_f[i]$ and $P_b[i]$, for forward and backward partial paths respectively, where i is the number of nodes included in the partial path. This is shown to ease the understanding of the algorithm, but from an implementation perspective, these partial paths have to be stored in a more efficient data structure such as a tree. A partial forward path tree keeps the source node

Data: A dynamic graph $G(V,E)$; the source node $s \in V$; the destination node $d \in V$; the starting time t_s; the maximum arriving time t_d; the transmission time τ; the minimum forward time t_f; the maximum backward time t_b; the forward-backward count c_{fb}; the set of partial forward paths P_f ; the set of partial backward paths P_b.
Result: A path P from s to d.
begin
  $c_{fb} = 0$;
  $t_c = t_s$;
  $P_f[c_{fb}] = \{\langle (s, t_s) \rangle\}$;
  $P_b[c_{fb}] = \{\langle (d, t_b) \rangle\}$;
  while $True$ **do**
    if $c_{fb} \geq 2$ **then**
      L_n = commonNodes($P_f[c_{fb}]$, $P_b[c_{fb}]$, G, τ);
      if $L_n \neq \emptyset$ **then**
        $P = getPathByNodes(P_f[c_{fb}], P_b[c_{fb}], L_n)$;
        return P;
      end
      L_e = commonEdges($P_f[c_{fb}]$, $P_b[c_{fb}]$, G, τ);
      if $L_e \neq \emptyset$ **then**
        $P = getPathByEdges(P_f[c_{fb}], P_b[c_{fb}], L_e)$;
        return P;
      end
    end
    $tmpForwardSet = \emptyset$;
    foreach $tmpForwardPath \in P_f[c_{fb}]$ **do**
      $tmpForwardSet =$
       $tmpForwardSet \cup extendForwardPath(G, tmpForwardPath, \tau)$;
    end
    $P_f[c_{fb}+1] = tmpForwardSet$;
    get the minimum forward time t_f;
    $tmpBackwardSet = \emptyset$;
    foreach $tmpBackwardPath \in P_b[c_{fb}]$ **do**
      $tmpBackwardSet = tmpBackwardSet \cup$
       $extendBackwardPath(G, tmpBackwardPath, \tau)$;
    end
    $P_b[c_{fb}+1] = tmpBackwardSet$;
    get the maximum backwardTime t_b;
    if $t_f > t_b$ *or* $c_{fb} \geq \left\lfloor \frac{L_{max}-2}{2} \right\rfloor$ **then**
      there is no path between s and d;
      return NULL;
    end
    $P_f[C_{fb}+1]$ = removeRedunantFpaths($P_f[C_{fb}+1]$, t_b);
    $P_b[C_{fb}+1]$ = removeRedunantBpaths($P_b[C_{fb}+1]$, t_f);
    $c_{fb} = c_{fb} + 1$;
  end
  return P;
end

Algorithm 3: Forward-backward path finder.

for the forward paths and all possible paths as successive branches. Each node stores the earliest time the node is visited. The same is analogously done for the partial backward nodes.

The intersection between forward and backward partial paths can be done looking for an intersecting node (yielding an odd path length) or edge (yielding an even path length).

Given a partial path $P' = \langle (v_1,t_1),\ldots,(v_l,t_l)\rangle$ of length l, we use $last(P') = (v_l,t_l)$ to denote the last node and associated time, and $first(P') = (v_0,t_0)$ to denote the first node and its associated time of the partial path.

The `CommonNodes` function, gets all common nodes that can be used to intersect forward and backward paths. The function will use P_f and P_b and try to compare the last node of partial paths in P_f, and the first node of partials paths in P_b to check if there are common nodes. That is, if $slast(P_f) = \{v_i | (v_i,t_i) = last(P'), \forall P' \in P_f\}$ denotes the set of last nodes from the set of partial paths P_f, and $sfirst(P_b) = \{v_j | (v_j,t_j) = first(P''), \forall P'' \in P_b\}$ the set of first nodes of partials paths in P_b, then $CommonNodes = slast(P_f) \cap sfirst(P_b)$.

Similarly, the function `CommonEdges` will attempt to find the path based on common edges of the last nodes from P_f and first nodes from P_b. More specifically, in this function it will check if there are some valid edges which satisfy the time requirement of the path. $CommonEdges = \{e = (s,d,t,\lambda) \mid e_i \in E, t+\lambda \geq timeP_f(s)+\tau, t \leq timeP_b(d)-\tau, \forall s \in slast(P_f), d \in sfirst(P_b)\}$. Where $timeP_f(v)$ returns the time associated to v in the corresponding partial path from the set P_f, and $timeP_b(v)$ the time associate to v in P_b.

Then, the function `getPathByNodes` or `getPathByEdges` will select a random node or edge from `CommonNodes` (L_n) or `CommonEdges` (L_e) respectively to construct the final path.

When the algorithm cannot get the path at the current iteration, it will use `extendForwardPath` and `extendBackwardPath` functions to increase the length of the partial paths both in P_f and P_b with all possible time-valid neighbors. The presence of loops is avoided by avoiding repeating a node when extending each path.

In order to reduce memory requirements the algorithm prunes partial paths from the forward and backward set. Paths which will not be possible to use for node or edge intersection are removed in each loop by the functions `removeRedundantFpaths` and

`removeRedundantBpaths`. Forward partial paths, where the time of the last node is greater than t_b, and backward partial paths, where the time of the first node is lower than t_f are removed. As described in the algorithm, t_f is the minimum time of the last node among all paths in P_f, and t_b is the maximum time of the first node from all paths in P_b.

3.2.4 Discussion

As expected, applying onion routing using the proposed path finding algorithms will enhance the communication privacy but will also lead to lower utility. In this case utility is understood as efficient communication in terms of path length, message delivery time, and even potential decrease in delivery ratios. It is difficult to quantify the utility loss and privacy gain in our scenario. To that end we have proposed several metrics to provide some measure of privacy and we also provide estimations of communication efficiency when compared to an optimal path finding algorithm. This will allow us to establish a measure of the trade-off between privacy and utility.

3.3 Privacy metrics

As OppNets are usually implemented as wireless network, we usually assume that the attacker can exploit this open access property to analyze the traffic in the whole network.

Determining the anonymity achieved by using onion routing with a given path will depend on several factors. Even with our stochastic approaches from the previous section, the topology and behavior of the network are key issues to take into considerations. In this chapter we will describe measures that can help in determining the anonymity achieved based on several aspects of the chosen path and the network topology and temporal behavior.

3.3.1 Anonymity set and anonymity degree

In order to estimate the anonymity in onion routing schemes it is quite common to rely on k-anonymity models [69] and entropy based metrics [70–72]. In our case we will assume two different and analogous adversarial models depending on the knowledge of the attacker. In general we assume that the attacker knows the dynamic behavior of

the network (which is usually public knowledge). Then, we consider two different cases, where the attacker knows:

1. The destination node and the time the message arrives to such node. In this case the goal of the attacker is to guess the source node of the communication.

2. The source node and the time when the message departed from such node. Now the attacker will attempt to guess the destination node.

The first model is the most commonly considered in generic onion routing schemes. There, the common scenario is for a given client to use TOR to connect to a given server. The client wants to hide its identity (IP address) from the server. In our case, we have considered both cases in terms of completion. The second case will be such that the attacker is observing (monitoring) the source node and will try to guess the destination of the message.

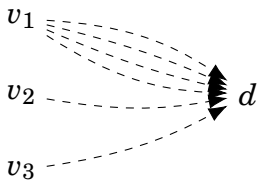
In this section we will present the definition of *anonymity set* and *anonymity degree* for the first model. The second one is analogous.

The *anonymity set*, $S(v,t)$, of a node $v \in V$ that has received a message at time t, is the set of all possible source nodes (possible origins of the message). In the general case, $S(v,t) \subseteq V$ for any v and t. The size of $S(v,t)$ can be used as an estimation of privacy or anonymity as it is done in typical k-anonymity models.

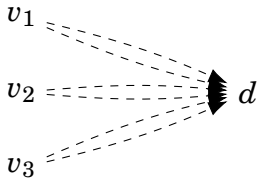
To complement this measure, we determine the entropy of the anonymity set, by considering the probability associated to each possible source node, which might not be uniform. We introduce the measure of such probability based on the number of existing simple paths that can reach the destination node from all possible source nodes. A *simple* path is a path that does not repeat nodes.

As an example, in Figure 3.1, we can see two different scenarios with an anonymity set of size 3 for the destination node d at time t_s, $S(d,t_s) = \{v_1, v_2, v_3\}$, the dashed arrows represent simple paths and we consider that all edges are present all the time to simplify the example.

We denote the set of simple paths from v_1 to d arriving at node d at time t_d as $P_{t_d}(v_1,d)$, so $|P_{t_d}(v_1,d)|$ is the number of different simple paths from v_1 to d. With this in



(a) $|S| = 3, H(S) = 1.25$



(b) $|S| = 3$, $H(S) = 1.58$

Figure 3.1: Example of the same anonymity set with different entropy.

mind, it is clear that the anonymity degree should be different in each case of Figure 3.1.

We define the probability of a given node v of being the source node of a path arriving at node d at time t_d as:

$$\mathscr{P}_{d,t_d}(v,d) = \frac{|P_{t_d}(v,d)|}{\sum_{u_i \in S} |P_{t_d}(u_i,d)|} \tag{3.1}$$

We can then compute the entropy of the anonymity set S for the destination node d at time t_d, detnoted as $S(d,t_d)$, as follows.

$$H(S(d,t_d)) = -\sum_{v_i \in S} \mathscr{P}_{d,t_d}(v_i,d) \log \mathscr{P}_{d,t_d}(v_i,d) \tag{3.2}$$

Following with the example, we have that $H(S(d,t_d))$ for Figure 3.1a is 1.25 while for Figure 3.1b is 1.58. Intuitively it is reasonable to have greater entropy for the second case, where the uncertainty about the origin is greater based on the possible paths to the destination node.

The maximum entropy of the anonymity set is achieved when $S = V$ and the probability for all $v_i \in S$ is uniform. Thus, we define the maximum entropy for the anonymity set as:

$$H_M = log(|V|) \tag{3.3}$$

Now we can introduce the *anonymity degree* $\mathscr{A}$ for a given destination node $d \in V$

and time t_d as:

$$\mathscr{A}(d,t_d) = \frac{H(S(d,t_d))}{H_M} \tag{3.4}$$

We are assuming here that the path is constructed following an stochastic approach and all paths from source to destination are equally probable.

The minimum possible value of the anonymity degree is 0, and will be achieved when there exists just one possible path ending at the destination node v at time t. In this case, the entropy of the anonymity set is 0, because there is no uncertainty about the source node. In turn, the maximum possible value of the anonymity degree is $1-\epsilon$ for a small ϵ, and it is reached when the entropy of the anonymity set is maximal, that is, when all nodes in the graph are valid sources and the probability of a path starting in each of them is exactly the same. Note that the entropy will never reach $\log(V)$, since the destination node v itself will never be a valid source node (we are requiring paths to be *simple* paths). As a consequence, the maximum anonymity degree tends to 1, but can not be exactly 1.

The anonymity degree informs about the privacy of a destination node at a given time in the context of a network of a certain order. Therefore, it can be used to compare the anonymity of nodes inside the same network or with nodes of other networks of the same order.

Although networks with high density are intuitively better from a privacy perspective, and will be so in the general case, we can not ensure that the anonymity degree in higher density networks will always be better. Think, for instance, in a very dense network where most of the paths end up at the same node.

On the contrary, if we only consider the size of the anonymity set, higher density networks will always present bigger (or equal) anonymity sets.

An important drawback of this anonymity measure is that its computation is not feasible for high density networks. Finding all simple paths has exponential memory requirements making it unfeasible for common desktop computers. We have used a time constrained modification of the Rubin algorithm [73] to compute all simple paths in relatively low density networks. For higher density networks we had to develop an approximated method.

The approximated method computes the approximated anonymity set, entropy of the anonymity set, and anonymity degree of a given destination node d which receives a message at time t_d by repeatedly searching backwards for simple paths ending at d. That is similarly as doing a time constraining reversed random walk repeatedly.

At each iteration of the algorithm, a new path ending at node d at t_d is searched by randomly selecting the previous hop of the path between the time backward neighbors (see Section 2.4) of the current node that have not been visited previously (i.e., that are not already part of the path).

At each hop of the path, the current time is updated and each of the iterations stops whenever one of the following conditions is reached:

1. the target number of hops is reached,
2. the maximum duration time is exceeded.

When the target number of hops of the path is reached, the iteration finishes and the last node is considered the source node of the path. On the contrary, when the maximum duration time is exceeded, the path is discarded. The approximated anonymity set of d at starting time t_d, $S'(d,t_d)$, is then the set of source nodes found during all the iterations of the algorithm.

Given an execution of the algorithm with n iterations, $p(v)$ is the number of times one of the iterations ends up in each source node v. Then, the estimated probability of node v being a source for a path ending at d at time t_d, $\mathscr{P}'_{d,t_d}(v,d)$, is $p(v)/n$.

Finally, the approximated entropy, H', and the approximated anonymity degree, $\mathscr{A}'$, can be computed with equations 3.2 and 3.4, using $\mathscr{P}'$ instead of $\mathscr{P}$.

On the one hand, notice that $|S'|$ will always be less than or equal to $|S|$, but this relation does not necessarily hold for $\mathscr{A}$ and $\mathscr{A}'$. On the other hand, the greater the number of iterations n of the algorithm, the more closer the results of the approximated algorithm should be with the exact ones.

We have seen how to compute S, $\mathscr{A}$ (and S', $\mathscr{A}'$) for the first adversarial model, where the attacker knows the destination node and the time that the message arrives to such node. From now on we will denote these measures as $\overleftarrow{S}$, $\overleftarrow{\mathscr{A}}$ (and $\overleftarrow{S'}$, $\overleftarrow{\mathscr{A}'}$). Analogous

measures of the anonymity set and the anonymity degree can be computed for the second adversary model. In this case the attacker knows the source node and the starting time and we can find all possible paths starting at such node in such given time. The definition of these measures is analogous to the ones we have seen and will be denoted from now on as $\overrightarrow{S}$, $\overrightarrow{\mathscr{A}}$ (and $\overrightarrow{S'}$, $\overrightarrow{\mathscr{A}'}$)

3.3.2 Path-degree measure

We have considered another metric based on the degree of all the nodes of a given path. In some sense these degrees give an estimation of the path uncertainty in each node. If an attacker knows a partial path, or can identify a given node in the path, the degree of the node and following unknown nodes can be seen as the difficulty in guessing the rest of the path. Moreover, the degree of a node in the path can be seen as the probability that an attacker has in guessing the next node of the path knowing only the current node and time.

Given a path $P = \langle (v_1, t_1) \ldots (v_l, t_l) \rangle$, of length l, its *path-degree* measure $\mathscr{D}$ is defined as:

$$\mathscr{D}(P) = \frac{1}{2}\left(\prod_{i=1}^{l}\left(deg^{t+}_{\mathscr{C}}(v_i, t_i, \delta_M)\right)^{-1} + \prod_{i=1}^{l}\left(deg^{t-}_{\mathscr{C}}(v_i, t_i, \delta_M)\right)^{-1}\right) \tag{3.5}$$

The path-degree measure combines the time forward and backward degrees of the nodes in the path in order to give a generic measure. The measure is defined in the interval $[0, 1]$. The value 1 is given by the worst case, where the degree of each node in the path, both forward and backward, is 1. On the contrary, values close to 0 are better from a privacy perspective since they denote higher degrees in the path.

For example, if we have a path $s \to r_1 \to r_2 \to r_3 \to d$. We can quantify the forward path degree from source node s to destination node d by considering the forward time neighbors of s, r_1, r_2, and r_3 at a given valid time determined by the time of each node in the path. If there are c_0 neighbors of s at the starting time, then the probability of choosing r_1 should be $\frac{1}{c_0}$. And the same applies to r_1, r_2 and r_3, respectively c_1, c_2, and c_3 neighbours according to different valid time. Thus, the forward path degree of the whole path from the source node to the destination node can be calculated as: $\prod_{i=1}^{l}\left(deg^{t+}_{\mathscr{C}}(v_i, t_i, \delta_M)\right)^{-1} = \frac{1}{c_0}\frac{1}{c_1}\frac{1}{c_2}\frac{1}{c_3}$. Similarly, we can compute the backward path degree from node d to node s using the same strategy.

The measure is not normalized with respect to the path length because we believe that such length should be taken into consideration. Higher lengths are better for privacy and will yield lower values for the path-degree measure.

3.3.3 Path Jaccard Distance

In the case of the random path finding algorithm we have considered the possibility of measuring how different are the paths found by the algorithm. That is, if we use the algorithm to repeatedly find a given number of paths, how different are those paths. They can be very similar, for example differing only in one node, or very different. The first case is obviously worst for privacy since it could lead to an attacker guessing the path with some minimum knowledge about it.

To that end, we use the Jaccard distance (JD) or Jaccard index to measure how different is one path from another. The formulation of the JD is as follows:

$$JD(S_1,S_2) = \frac{|S_1 \cap S_2|}{|S_1| + |S_2| - |S_1 \cap S_2|} \tag{3.6}$$

S_1 and S_2 are the two different sets, we note the value of the JD of S_1, S_2 as $JD(S_1,S_2)$. The value of $JD(S_1,S_2)$ belongs to $[0,1]$, where, if S_1 and S_2 are the same sets, it gives 1, and if the sets are completely different it will give a value of 0.

In our case, we consider each path as a set of nodes and compute the Jaccard index between such sets. This is obviously an approximate measure since it does not take into account the order of the nodes, but on the contrary it provides a measure, which can be computed very efficiently, and that we believe can closely estimate the idea that two paths are similar or different.

In our experiment for the next following chapter, the results of the JD between paths selected show that the possible paths found in our scenario are quite different. We obtain values very close to 0.

3.3.4 Discussion about the privacy measures

When considering privacy in our proposal, the anonymity set size measure can be comparable to the well known k-anonymity model. The related anonymity degree helps in better understand the actual distribution of probabilities within such anonymity set. It is important to note that we need to provide both measures together, given that the

anonymity degree is given as a ratio of the maximum possible entropy. As an example, we can have two different anonymity sets with the same anonymity degree but with very different sizes. In such case, clearly the case with the bigger set is better for privacy. When the size of the anonymity set is the same, the cases with higher entropy are better for privacy. In some sense, the anonymity degree can be seen as a measure of diversity related to the anonymity set, similarly as l-diversity [74] is related to k-anonymity. It gives information about the distribution of probabilities within the anonymity set.

On the other hand, the path-degree measure, gives information about how easy or difficult it might be to obtain a given path by a random walk (forward and backwards). It gives information about the diversity of alternatives to follow in the path. They main idea is to summarize in a very broad and general manner, the possible difficulties that an attacker observing a partial path can have to complete the whole path.

Both types of measures are not directly related since they measure different things. The anonymity set and anonymity degree are possibly the most interesting ones, and the ones that have more relation to how anonymity is traditionally measured. We think however that the path-degree measure is an interesting complement to the other ones.

As we will observe in the evaluation (see Chapter 5), in general, all privacy measures are highly related to the density of the network. Higher density will yield better privacy measures.

3.4 Evaluation

In this section we present the evaluation of our proposal and the results we obtain. First, we introduce the dataset and setup of the experiments, and then the results regarding performance and anonymity.

3.4.1 Dataset

In our experiments we have used the CRAWDAD *rice/ad_hoc_city* dataset [75]. This is a wireless ad hoc opportunistic network based on the Seattle public bus transportation. Each network node is located in a bus and contacts between them are established based on the coverage of the two nodes (ability to send a message between them). It can be considered as a POppNet due to the high predictability in bus timetables and itineraries. The network contains close to 1200 buses covering a 5100 square kilometer area. It is

clear that there will be errors in the predictability of the network but we assume these errors to be tolerable in the application domain (e.g. the use of our proposal in live critical communication systems is completely discouraged). This is something usually assumed and accepted in opportunistic networking. Transmission time is assumed to be 1 second (this includes establishing the connection and sending a short message).

Network	$\|V\|$	$\|E\|$	avg. degree	graph density
N1	1179	543692	922.293	0.783
N2	1179	271813	461.091	0.391
N4	1177	135976	231.055	0.196
N8	1178	67896	115.273	0.098
N16	1170	34090	58.274	0.050
N32	1157	17064	29.497	0.026

Table 3.1: Datasets

The network that we consider spans through 24 hours, and has a high density with more than $500,000$ dynamic edges during this time interval. Network density is a key issue in our proposal. Higher density means it is easier to construct paths and more difficult for an attacker to predict possible potential paths. In order to test our results with lower density networks we have produced different versions of the same Seattle network with lower number of edges. More precisely, we have obtained networks $N2$, $N4$, $N8$, $N16$, $N32$, by randomly selecting 1/2, 1/4, 1/8, 1/16, 1/32 number of edges of the original network denoted as $N1$. Table 3.1 shows the number of edges, average degree, and graph density for each network taken from their corresponding static graph G^*. Given that they correspond to the same network, with the same number of nodes and the same overall behavior it makes the comparison between them to be focused exclusively on the network density.

We also note that the distribution of the edges overtime is not uniform, as expected in a real network based on a public transportation system. There are hours with higher density than others, presumably corresponding to rush hours as shown in Figure 3.2.

We have performed our experiments in different time frames. We consider two types of starting times for our experiments. The experiments denoted as *zero* are paths that start at a random time from the interval $[0, 10000]$, while experiments denoted as *rush* are starting in the interval $[20000, 35000]$.

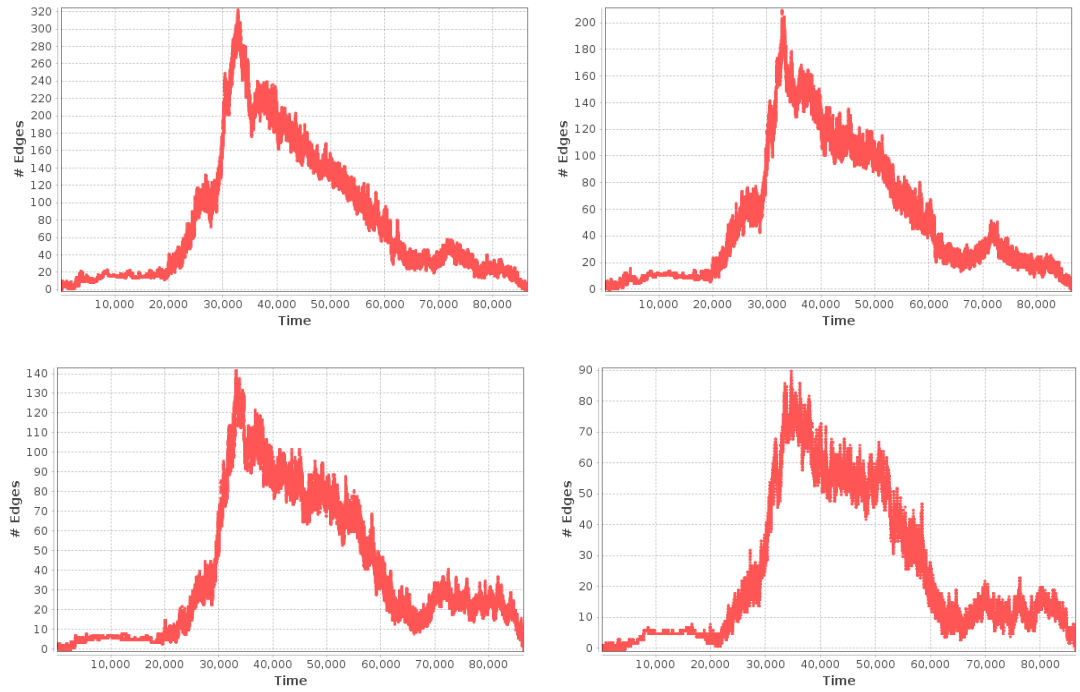


Figure 3.2: Distribution of edges for networks $N1$, $N4$, $N16$, and $N32$ over 24 hours (time given in thousands of seconds).

The performance experiments from Section 3.4.2 have been executed in a desktop computer, Intel i7 CPU at 3.4GHz, 16GB memory. For the exhaustive computation of the anonymity degree (cf. Table 3.4) we used a computer with an Intel Xeon E3-1230 V2 at 3.30GHz, and 30 GB of memory.

Given the stochastic nature of our proposed algorithms, and the different scenarios that can happen in real networks, each experiment is composed of 100 cases for *zero* and *rush* starting times, giving a total of 200 executions. For each case a different pair of nodes (source and target), and starting time are randomly selected, then the average is usually considered. We also separate *zero* and *rush* experiments in order to be able to appreciate different situations.

We will denote our random algorithm as R (Section 3.2.2), and the forward-backward algorithm as FB (Section 3.2.3). In order to properly evaluate our proposal, we need to consider the penalty introduced by using our algorithms. That is, what do we have to sacrifice in terms of performance as compared to a non-anonymous routing. For such purpose we denote the shortest path algorithm from Section 3.2.1 as X. As we previously noted this algorithm is deterministic and thus, not desirable for anonymous

communications.

We have conducted experiments both in terms of performance and privacy. Performance experiments show how good are the stochastic algorithms in terms of their efficiency, while privacy experiments evaluate their security.

3.4.2 Performance

We show here the performance evaluation of finding paths using the algorithms from Section 3.2 by evaluating several parameters or characteristics and usually comparing them to the results obtained with the shortest path algorithm X. The X algorithm can be considered in most cases the best performance achievable.

3.4.2.1 Applicability related to network density

To be applicable, our algorithms need to be able to find a number of paths from source to destination. An algorithm can fail in finding a path from source to destination either due to the nonexistence of this path or because it is just not able to find an existent path due to its stochastic nature. Not only we need to find a path but a minim number of paths big enough to provide good anonymity. We compare the behavior of R, and FB algorithms to that of X, and related to the density of the network in Figure 3.3. The failure rate indicates the average percentage of paths not found on each network. Clearly, if a path exists between two nodes X will provide such path, while R, and FB might fail.

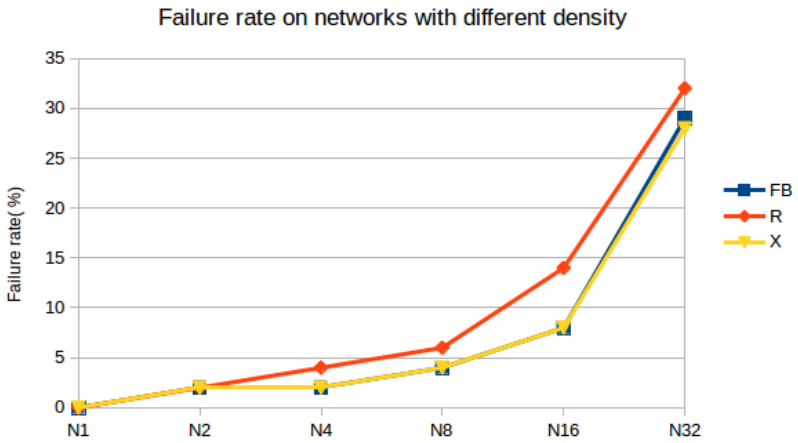
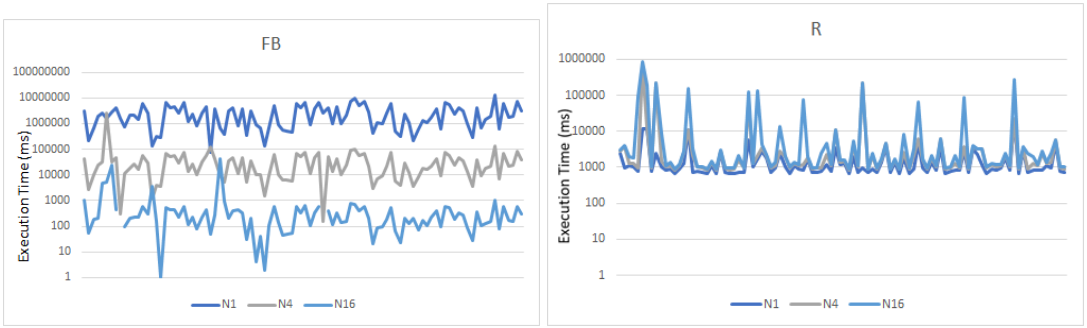


Figure 3.3: Applicability of the algorithms on networks with different density.

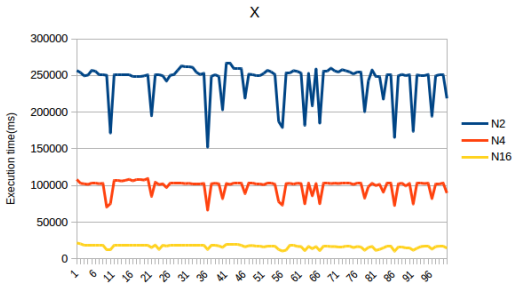
We can see that any algorithm can be successfully applied more than 95% of the times on networks with high density ($N1$ to $N4$), while, when density decreases, the behavior of FB is very close to that of X and thus is preferable over R. FB can still be successfully applied more than 90% of the times on $N8$ and $N16$. However, on $N32$, which has a very low density, the applicability of any algorithm sharply decreases.

3.4.2.2 Execution time



(a) Execution time for FB.

(b) Execution time for R.



(c) Execution time for X.

Figure 3.4: Average execution time

Figure 3.4 shows the execution time of X, R, and FB on networks with different density. In general our algorithms R and FB are faster than X, notably R. As expected, execution time is proportional to the network density, although the difference is less appreciable in R.

3.4.2.3 Path length

The minimum path length should be always 5, since it is the minimum path length required for onion routing to be secure. Even so, larger paths can be found by R and FB given their stochastic behavior. We compare the results of R and FB with the actual shortest path obtained from X.

Net	*FB*	*R*	*X*
N1	5.0 (0.0)	6.8 (1.4)	4.0 (1.0)
N2	5.0 (0.0)	6.9 (1.5)	4.0 (1.0)
N4	5.0 (0.0)	7.1 (1.6)	4.0 (1.0)
N8	5.1 (0.6)	7.2 (1.7)	4.2 (1.7)
N16	5.2 (0.9)	6.8 (1.5)	5.0 (1.0)
N32	5.5 (1.3)	7.1 (1.3)	5.8 (2.5)

(a) Zero time

Net	*FB*	*R*	*X*
N1	5.0 (0.0)	6.8 (1.0)	3.5 (1.1)
N2	5.0 (0.0)	6.7 (1.2)	3.7 (1.0)
N4	5.1 (0.4)	7.1 (1.7)	3.7 (1.1)
N8	5.1 (0.6)	7.3 (1.7)	4.4 (1.5)
N16	5.2 (0.9)	7.1 (1.2)	4.5 (1.5)
N32	5.5 (1.3)	7.2 (1.6)	5.0 (1.8)

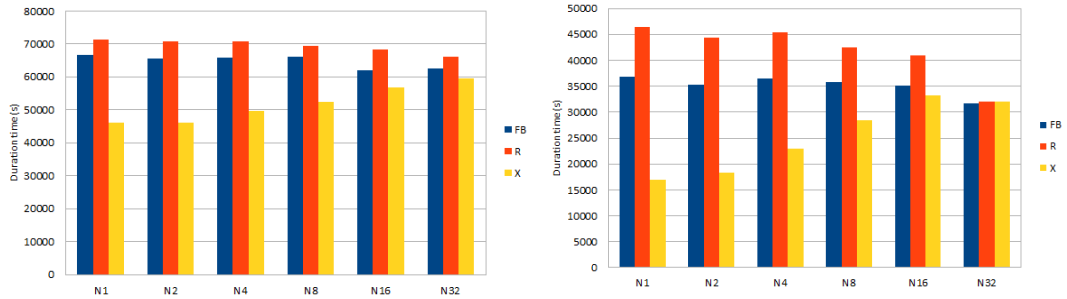
(b) Rush time.

Table 3.2: Average path length. Standard deviation is shown in parenthesis.

Table 3.2 shows the average path length for each execution of FB and R and the shortest path, X. FB obtains paths closest to the required minimum due to its meet-in-the-middle strategy. Paths from R are larger but within a reasonable range.

3.4.2.4 Duration time

Figure 3.5 shows the path duration time (c.f. Section 2.4) for all the algorithms and networks in zero and rush times.



(a) Zero time

(b) Rush time

Figure 3.5: Duration time

Shortest duration times are obtained in rush time experiments, and in general FB paths have a shorter duration than R. In any case we consider the penalty in duration time, as compared to X, to be quite acceptable for our scenarios.

3.4.2.5 Path Jaccard distance

Figure 3.6 depicts the path Jaccard distance of the possible paths found with the random path finding algorithm R. For each pair of nodes, we compute 100 different paths. We

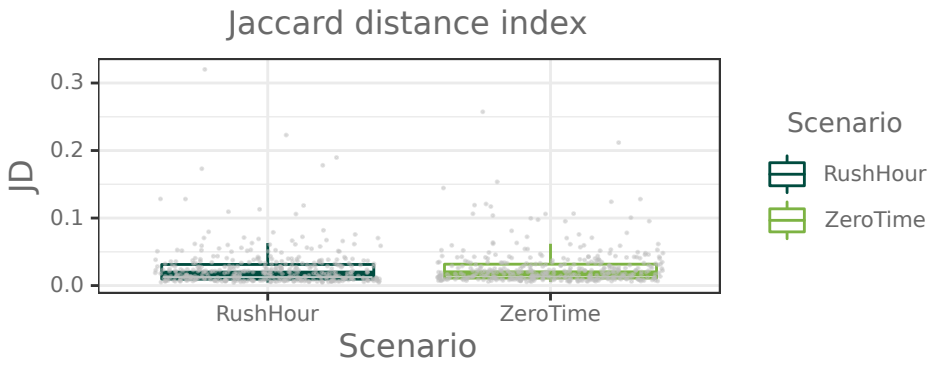


Figure 3.6: Jaccard distance index

then compute the path Jaccard distance for each pair of paths, and we get the average value of all these Jaccard distance indexes. We can see from the results that the paths are quite different which makes it difficult for a potential attacker to guess the correct routing path.

3.4.3 Anonymity

In order to evaluate the privacy achieved by our proposal we rely in the measures described in Section 3.3.

3.4.3.1 Anonymity set and degree

As mentioned in Section 3.3.1, the anonymity set size and the anonymity degree can be difficult to compute in high density networks. We can however use the approximate method also introduced in Section 3.3.1. We have computed the anonymity set size and anonymity degree for both adversarial scenarios from Section 3.3.1, that is $|\overleftarrow{S}|$, $\overleftarrow{\mathscr{A}}$, and $|\overrightarrow{S}|$, $\overrightarrow{\mathscr{A}}$, and their respective approximated values.

Table 3.4 shows the exact anonymity degree and anonymity set size for $N32$, and $N16$. Those are the only networks where we could compute the exact values with our equipment (cf Section 3.4.1). On the other hand, Table 3.6 shows their approximation using the approach described in Section 3.3.1. The results where obtained using a fixed path length of five nodes and fixing the maximum duration of the path to the average of the cases obtained from algorithm R as detailed in the previous section. One million iterations of the algorithm were performed for each analyzed source node.

Net	$\lvert\overleftarrow{S}\rvert$	$\overleftarrow{\mathscr{A}}$
N32	834.47 (228.5)	0.7969 (0.16)
N16	1045.61 (120.5)	0.8703 (0.10)

(a) Zero hours

Net	$\lvert\overleftarrow{S}\rvert$	$\overleftarrow{\mathscr{A}}$
N32	712.36 (316.8)	0.7632 (0.21))
N16	1022.83 (141.6)	0.8717 (0.11)

(b) Rush hours

Table 3.3: Exact anonymity set size ($\lvert\overleftarrow{S}\rvert$) and degree ($\overleftarrow{\mathscr{A}}$) for N32, N16, N8. Standard deviation is shown in parentheses.

Net	$\lvert\overrightarrow{S}\rvert$	$\overrightarrow{\mathscr{A}}$
N32	923.66 (272.4)	0.8229 (0.14)
N16	1096.6 (154.85)	0.8703 (0.10)

(a) Zero hours

Net	$\lvert\overrightarrow{S}\rvert$	$\overrightarrow{\mathscr{A}}$
N32	822.03 (344.2)	0.7855 (0.22)
N16	1038.69 (226.6)	0.8551 (0.12)

(b) Rush hours

Table 3.4: Exact anonymity set size ($\lvert\overrightarrow{S}\rvert$) and degree ($\overrightarrow{\mathscr{A}}$) for N32, N16, N8. Standard deviation is shown in parentheses.

Net	$\lvert\overleftarrow{S'}\rvert$	$\overleftarrow{\mathscr{A}'}$
N32	801.19 (222.01)	0.854596 (0.12)
N16	951.89 (97.27)	0.904594 (0.03)
N8	1004.04 (81.24)	0.912801 (0.03)
N4	1025.11 (65.5)	0.916637 (0.02)
N2	1036.66 (46.6)	0.918900 (0.02)
N1	1043 (46.93)	0.919998 (0.02)

(a) Zero hours

Net	$\lvert\overleftarrow{S'}\rvert$	$\overleftarrow{\mathscr{A}'}$
N32	566.5 (271.46)	0.746111 (0.23)
N16	849.6 (127.59)	0.875215 (0.05)
N8	930.81 (77.11)	0.895185 (0.03)
N4	964.9 (68.14)	0.901639 (0.03)
N2	969.28 (62.61)	0.902211 (0.03)
N1	989.09 (63.57)	0.905207 (0.03)

(b) Rush hours

Table 3.5: Approximated anonymity set size ($\lvert\overleftarrow{S'}\rvert$) and degree ($\overleftarrow{\mathscr{A}'}$). Standard deviation is shown in parentheses.

We consider the approximated approach to be very accurate for the anonymity degree. The estimation error is given in Table 3.7 for both adversary models The estimation error on the anonymity set size is a bit bigger, but we consider it also to be acceptable (see the relative error from Table 3.7). This is due to the fact that finding all possible paths with the approximated approach is difficult. This approach will more likely find destination nodes with higher number of paths, which makes the anonymity degree more accurate that the anonymity set approximations.

Note that the results of both privacy measures (namely, the anonymity set size and the anonymity degree) increase with the density of the network. Moreover, in this scenario,

Net	$\lvert\overrightarrow{S'}\rvert$	$\overrightarrow{\mathscr{A}'}$
N32	792.88 (297.7)	0.825724 (0.2)
N16	970.88 (253.8)	0.889494 (0.12)
N8	1027.2 (236.74)	0.908159 (0.07)
N4	1061.1 (203.28)	0.920988 (0.05)
N2	1067.69 (196.53)	0.922615 (0.04)
N1	1074.93 (195.42)	0.924665 (0.04)

(a) Zero hours

Net	$\lvert\overrightarrow{S'}\rvert$	$\overrightarrow{\mathscr{A}'}$
N32	565.77 (304.9)	0.736670 (0.26)
N16	852.38 (292.27)	0.842796 (0.18)
N8	966.36 (241.81)	0.891140 (0.09)
N4	1020.53 (221.87)	0.909172 (0.05)
N2	1027.16 (218.55)	0.909134 (0.06)
N1	1042.02 (204.87)	0.913349 (0.05)

(b) Rush hours

Table 3.6: Approximated anonymity set size ($\lvert\overrightarrow{S'}\rvert$) and degree ($\overrightarrow{\mathscr{A}'}$). Standard deviation is shown in parentheses.

utility can be understood as the ability to successfully and efficiently send messages. Therefore, utility also increases with the density of the network (see Section 3.4.2): when more edges are kept in the network, it is more probable to find a valid path that can be used to send a message. As a consequence, in this scenario privacy and utility go hand by hand.

Net	Time	$\lvert\overleftarrow{S}\rvert$ error		$\overleftarrow{\mathscr{A}}$ error	
		absolute	relative	absolute	relative
N32	Zero	33.28	0.0399	0.0577	0.0724
	Rush	145.86	0.2048	0.0171	0.0224
N16	Zero	93.72	0.0896	0.0343	0.0394
	Rush	173.23	0.1694	0.0035	0.0041

Net	Time	$\lvert\overrightarrow{S}\rvert$ error		$\overrightarrow{\mathscr{A}}$ error	
		absolute	relative	absolute	relative
N32	Zero	130.78	0.1416	0.0028	0.0034
	Rush	256.26	0.3117	0.0488	0.0621
N16	Zero	125.72	0.1146	0.0192	0.0220
	Rush	186.31	0.1794	0.0123	0.0143

Table 3.7: Estimation error in the approximated calculations.

We can also compute the path degree measure from Section 3.3.2 using the paths found in Section 3.4.2. As an example, the averages of all cases for each network and time interval for paths found with the FB algorithm are given in Table 3.8.

We can see that, although the path-degree measure increases as the network density

	$\mathscr{D}(P)$	
Network	Zero	Rush
N1	$1.26E-20$	$1.22E-18$
N2	$4.65E-19$	$9.60E-18$
N4	$2.96E-16$	$1.78E-15$
N8	$2.89E-15$	$2.82E-14$
N16	$4.94E-11$	$3.26E-09$
N32	$6.10E-09$	$5.00E-14$

Table 3.8: Path-degree measure

decreases, the values for the lower density network are still very low. This measure gives an idea of the degree of each node in the path, or the difficulty for an attacker of guessing the next node (given a concrete node on the path) in both directions, forward and backward.

3.5 Conclusions

In this chapter we have presented the applicability of Onion Routing to POppNets. We have introduced two different approaches to determine the onion routing paths introducing randomness in the path selection. We also introduced different metrics to measure some privacy related characteristics of the path and specific scenarios. In this regard we have considered an specific network, which is taken from a public bus transportation network. Focusing in a realistic scenario allows us to validate our results. It is clear that a different scenario might produce different results which might not be so good. We believe however that we have provided enough mechanisms to evaluate the convenience and suitability of our approach to specific scenarios. For instance networks with low density will produce worst results regarding both performance and privacy. This gives an estimation of the applicability of our proposal.

CHAPTER 4

ANONYMOUS COMMUNICATION IN OPPNETS WITH MIX NETWORKING

When considering general OppNets, as opposed to the POppNets from the previous chapter, the application of mechanisms based on onion routing becomes more complicated. Although these could be interesting approaches, some scenarios of OppNets need to address the problem of traffic analysis. Contrary to the use of onion routing in the Internet for example, in some OppNets scenarios having and attack model where the attacker has access to the whole communication medium is not rare. Typical OppNets use wireless connections where traffic analysis imposes a dangerous threat. Attacks can exploit the message flow to detect the whole onion routing path, which will leak the communication pattern between sender and receiver, assuming that the whole network traffic is not extremely high. A common approach to deal with this problem is to generate dummy network traffic to mask the actual traffic in the form of noise. Another alternative, which has not been very investigated, is the us of Mix networks (Mixnets). We propose in this paper the use of a Mixnet mechanism to provide privacy in OppNets.

As we will show, Mixnets can be used to provide anonymous communications in OppNets without incurring in a high impact on the normal operation of the network. Some nodes from the network will act as Mix nodes enabling the implementation of different strategies. To the best of our knowledge, our anonymous schema is the first work to propose the use of Mixnets in OppNets. Our main contributions are the followings:

Firstly, we design the Mixnet-based communication approach for OppNets. Secondly, we validate and show the particularities of our proposed schema in the ONE simulator (a common network simulator specially focused towards OppNets). Thirdly, we have analyzed the performance and the privacy property of our proposed mechanism.

4.1 Opprtunistic Mix networking

Our proposal is based on the use of a Mixnet in an OppNet. The main idea is to be able to use a relatively simple Mixnet schema, without requiring complex cryptographic mechanisms. As we will see, this approach not only is feasible in some OppNet scenarios but can also set the base for interesting solutions to anonymous communications in dynamic networks.

In order to implement the Mixnet, each network has a set of m Mix nodes $\mathscr{M} = M_1,\ldots,M_m$, such that $\mathscr{M} \subseteq V$. Our proposal will exploit the fact that there could be several Mix nodes, and the source node is free to choose which ones it uses to send a message. Moreover, the sender can also choose the number of Mix nodes to use in a cascade fashion (network) from the set $\mathscr{M}$. In this sense our approach follows what could be denoted as a restricted *free route Mix network*, it is a free route Mixnet [76, 77], but the selection of nodes is limited to a subset of the nodes of the network. We will assume that there is a key distribution mechanism so each Mix node has an asymmetric key pair, and the public key of each Mix node is known to all nodes.

We will consider a typical decryption Mixnet. For example, consider a sender node V_s that chooses three Mix nodes M_1, M_2, M_3 to send the message x to the destination V_d. Each node has a corresponding public key $PK_s, PK_1, PK_2, PK_3, PK_d$. To that end V_s builds an onion-based encryption scheme such as:

$$PK_1(r_1, PK_2(r_2, PK_3(r_3, PK_D(r_d, x))))$$

Where r_i are random nonces ensuring that the messages can not be correlated to their encrypted versions in any step. The message is received by M_1, decrypted, and send it to M_2 after performing the Mix. Each node does the same until the message arrives at the destination V_d. Each node V_i waits until it has received k_i messages, and then forward these k_i messages, making it hard to an attacker to correlate input messages with output messages.

In the general case, the Mix nodes are a subset of the whole network nodes, $\mathscr{M} \subseteq V$. The source node selects a path of Mix nodes $M_i \in \mathscr{M}$ of length p, and encrypts the message x as:

$$PK_1(r_1,\ldots PK_p(r_p,PK_D(r_d,x))\ldots)$$

We assume that each Mix node has enough capacity to perform the cryptographic operations and memory to store the required k_i messages. Which nodes act as Mix nodes will depend on the specific scenario, and the parameterization of our approach. If we assume that we can choose such nodes, it is important to note that this can have implications in several ways. It could seem that having more Mix nodes will provide more privacy. The sender can choose randomly from the set of Mix nodes and thus make it harder for an attacker to follow the message through the network. On the other hand, having a lot of Mix nodes can make the delay associated to each Mix bigger, since each node will need to wait for k_i messages. We can also consider other approaches to selecting potential Mix nodes, for instance it seems reasonable that those nodes with more interactions will obviously be more interesting to use. We will discuss this in Section 4.3.

4.2 Message routing

Routing messages in our opportunistic Mixnet approach differs greatly from a common Mixnet implemented in a more classical network. Messages might not be easily routed and source routing cannot generally be used. Thus, when the source node chooses the path of Mix nodes, delivering the message to such nodes might require routing it through other intermediary nodes. To that end our proposal uses *epidemic routing*.

In epidemic routing, each node forwards a message to all its neighbours until the message reaches its destination. It can produce a lot of flooding in the network and, although it is not the most efficient routing strategy for OppNets, it is the most generic an basic approach and serves us as a good base to evaluate our proposal.

As an example, Figure 4.1, shows a possible message delivery from the source node V_s to the destination node V_d. The source node chooses three Mix nodes M_1, M_2, M_3. The delivery of the message needs to use intermediary nodes denoted as V_i for $i = 1,\ldots,8$. The dashed lines denote epidemic routing, which is used to deliver the message from V_s to M_1, then from M_1 to M_2, and so on. We denote these routes between each Mix node and source and destination nodes as *routing steps*. The number of intermediate

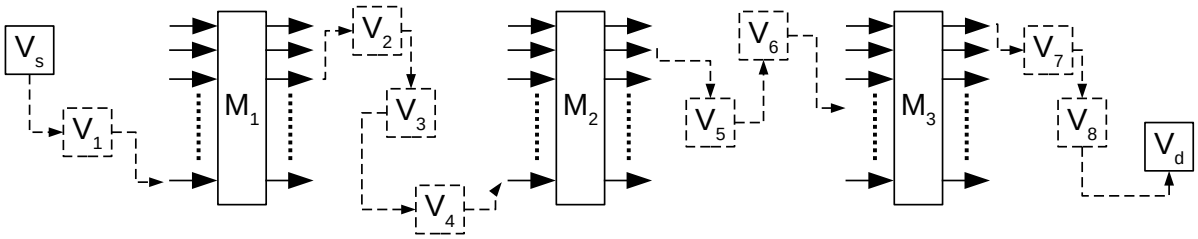


Figure 4.1: Example of opportunistic Mixnet.

nodes used in each routing step will depend on the network behaviour at the moment. As described before, each Mix node will have to wait until receiving k_i messages, which will introduce an additional penalty to message delivery time.

4.3 Choosing Mix nodes

As described before, the localization of Mix nodes in the network is an important issue. In order to evaluate our opportunistic Mixnet proposal, we will choose a set of network nodes $\mathscr{M}$ which can act as Mix nodes. The size of $\mathscr{M}$, denoted as $|\mathscr{M}|$, or more precisely its size with respect to $|V|$ will determine the overall performance of the Mixnet. A part from its size, another important decision is how to choose such set $\mathscr{M}$. There are existing routing strategies in free route Mixnets to optimize the anonymity of the system [78]. These strategies however assume that all nodes are reachable and routing does not occur opportunistically. In order to evaluate the use of our proposal we have opted to provide two selection strategies focused on the performance of the opportunistic Mixnet.

That is, we use two different strategies to select the set $\mathscr{M}$:

- **Random selection:** $\mathscr{M}$ is randomly selected among all nodes V. This is a relatively simple idea where the nodes acting as Mixes can be any node from the network. No other characteristic is taken into account.
- **Centrality-based selection:** $\mathscr{M}$ is selected as the $|\mathscr{M}|$ nodes from V with higher centrality. Here the idea is to promote the use of highly connected nodes to act as Mixes. This should improve the performance on the overall network. In order to simulate this idea we look at nodes with higher centrality. As centrality measure we use the historic number of neighbours of each node. That is, a node with more contacts over time will have higher centrality. More precisely the centrality of a node $v \in V$ is denoted as $C(v) = |N^*(v)|$. In a real situation this could be equivalent

to select nodes which we know that will have higher centrality or connectivity. E.g. in some vehicular networks, road side units are known by every member of the network and they are central nodes also.

Intuitively, the centrality-based strategy will yield better delivery ratios and better delivery time. Those nodes will have better connectivity and messages will be more efficiently routed among them.

The size of the set $\mathscr{M}$ with respect to V has also performance and privacy implications but such implications might not be that obvious. For instance a big set of potential Mix nodes will imply:

- Increase privacy: as more nodes can be used as Mix nodes, the attacker will need to monitor more nodes attempting to correlate inputs with outputs. This introduces confusion as a mean of privacy. It is important to note that privacy is actually determined by the number of messages that each Mix node accepts, and the number of Mix nodes each message uses. The fact that there are more Mix nodes present in the network hiders the task of the attacker in monitoring such nodes and can provide a better privacy due to the distribution of links in the whole Mix network [78].

- Introduce more time delays: as there are more Mix nodes that can be used, filling each node with its corresponding k_i messages will take more time and the delay introduced by each Mix node is expected to be higher. This will obviously depend on the actual traffic of the network (number of messages exchanged over time), but in the general case, more Mix nodes will produce higher delay times.

4.4 Threat model

The objective of the adversary is to link the sender and receiver of a message in the OppNet. Specifically, we assume that the attacker has the ability to access the whole communication network. It can wiretap, forward and delete messages. The attacker can also compromise the user and Mix nodes in the network. However, we assume that there is at least some honest Mixes.

Our threat model is based on [79]. As we all know, there is no prefect security defender that could resist all kinds of attacks. In our assumption, we mainly consider three different threats.

- **Traffic analysis.** An attacker A can conduct a traffic analysis to find the link between the sender node S and the receiver node D. A has the capacity to get access to every single message M and its goal is to speculate that S and D are communicating with each other with a high probability.
- **Compromised user nodes.** We also assume that the attacker A can compromise the user nodes. Thus, A can obtain the message M and forward it.
- **Compromised Mix nodes.** The Mix nodes can also be compromised but at least one of them should be honest. The compromised Mix nodes can conduct the link attack to deanonymize the system.

4.5 Simulation and results

We present an experimentation using an enhanced version of the Opportunistic Network Environment (TheONE) simulator [80], that includes our Mixnet anonymous routing. We have chosen different scenarios to analyse the performance of our schema. Node contacts from the scenarios are defined by physical contacts obtained from real mobility traces from the Crawdad database[1], a community resource for collecting wireless data at Dartmouth College. Each scenario corresponds with a different network, with different nodes and different mobility and contact patterns.

The first scenario, the *Info5* scenario, is based on real mobility traces [81] obtained during the 2005 edition of the Infocom conference in the course of almost 3 days. Contacts from these mobility traces represent 22459 contacts from 41 different nodes. The second scenario, the *Cambridge* scenario, is based on 10641 real contact traces from 51 students from the System Research Group of the University of Cambridge carrying small devices for six days [82]. Finally, the third scenario, *Taxis* [83], contains 449226 mobility contacts from 304 taxis during one month in the city of Roma.

Table 4.1 summarizes the main characteristics of the three scenarios. These include the *connections per minute*, which is the average number of connections a network has

[1] *http://crawdad.org/*.

	Scenarios		
	Cambridge	Info5	Taxis
Number of nodes	51	41	304
Connections per minute	0.9979	8.035	15.13
Total connection time (sec.)	1383082	2145974	6189716
Max node degree	12	14	12
Average node degree	0.2996	0.6433	0.4713

Table 4.1: The characteristics of the 3 used scenarios.

in one minute during the whole duration of the scenario; *total connection time* the sum of the duration of all connections in the network, $\sum_{e_i \in G} \lambda(e_i)$; and the maximum and average degree of all nodes $v \in V^*$, where the degree of the node is determined by $N^*(v)$ (see Section 4.1).

These scenarios are commonly used in the evaluation of routing algorithms in the Oppnet literature. They provide a realistic setup to measure the feasibility of our proposal.

In our experiment, we set the simulation time of 86400 seconds (24 hours). That is, we simulate each scenario during 1 day.

4.5.1 Performance metrics

One of the main concerns when using not only Mixnets but any means to provide anonymous communications in a network, is the penalty that such solution could introduce worsening the overall performance on the network. This is specially relevant in Oppnets where we already do not usually have good performance in message delivery. The use of Mix nodes will introduce latency in message delivery and decrease the delivery ratio. The goal is to see if these penalties are tolerable or not. To that end we have measured message latency and delivery ratio in several simulations, for the three proposed scenarios and using different setups and parameterizations.

We have mainly focused our experiments to observe:

- **Message latency:** the average time it takes for a message to reach its destination.
- **Delivery ratio:** the ratio of successfully delivered messages.

- **Average routing step path length:** the average number of hops that one packet needs to traverse in each routing step using epidemic routing (see Section 4.2). That is, the number of nodes used to route a message from the source node to the first Mix node, from the first Mix node to the second one, and so on.

There are two main parameters that influence how our proposal is deployed in a given scenario. One is the **percentage of Mix hosts**, i.e., the percentage of nodes can be chosen as Mix hosts or the size of $\mathscr{M}$ with respect to V. Those are $20\%, 40\%, 60\%, 80\%, 100\%$.

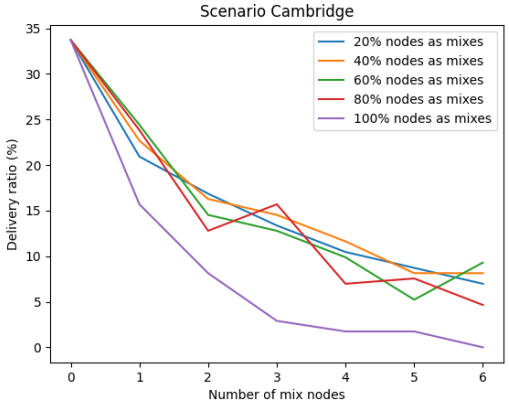
The other parameter is the **number of Mix nodes** used in cascade. That is, the number of Mix nodes that each message will go through. In our experiment, we set the **number of Mix nodes** from 0 to 6. Here the value set to 0 means not using any Mix node. In this case, routing is performed using epidemic routing from the source to the destination. To simulate a real OppNet environment, the TTL of each message set to a relative large value of 5.5 hours (20000 seconds). We assume that each node has enough buffer to store the messages. During the simulation, every 100 seconds, a randomly selected node sends a message of size 1 byte to a randomly chosen node. This is performed in all three scenarios. And the parameter $k_i = 10$ for each Mix node i mentioned in previous sections.

4.5.2 Results

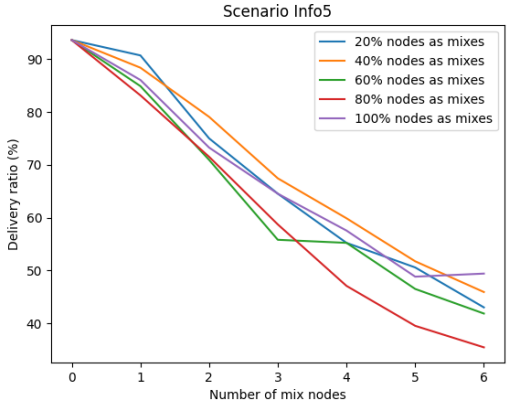
Running the simulations we can measure the penalty introduced by using the opportunistic Mixnet approach in the three above mentioned scenarios.

Figure 4.2 shows the delivery ratio for each scenario. In each case the figure shows the delivery ratio based on the number of nodes used in cascade to communicate. Note that 0 Mix nodes corresponds to the case of not using our opportunistic Mixnet approach. Each line corresponds to a different size for the set $\mathscr{M}$ of Mix nodes. We use sizes corresponding to the 100%, 80%, 60%, 40%, and 20% of the total nodes. This size is selected randomly from the whole set of nodes of the network. Similarly Figure 4.3 shows the same data but the selection of the set of Mix nodes is performed using the node centrality.

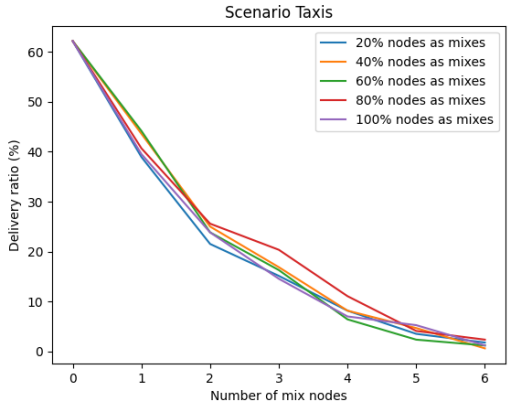
We can observe that in general the selection of nodes with higher centrality as Mix nodes provides slightly better results in terms of delivery ratio. This improvement is however relatively small in most cases.



(a) Cambridge



(b) Info5

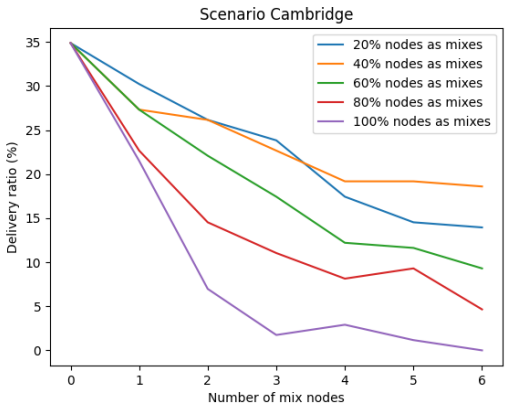


(c) Taxis

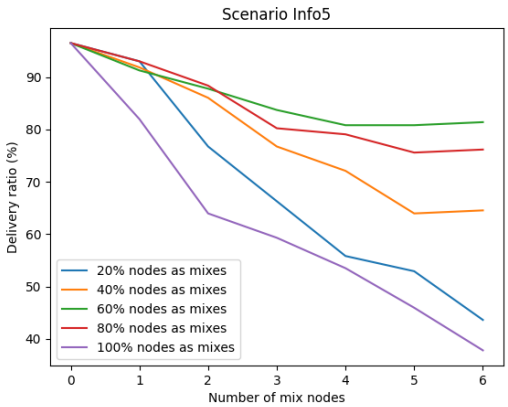
Figure 4.2: Delivery ration with random Mix nodes.

We also note that using a higher number of nodes as Mix nodes, that is higher size for the set $\mathscr{M}$, usually provides lower delivery ratios. This is due to the fact that increasing the number of Mix nodes also increases the delay introduced by each one of them. Recall that the source node selects the Mix nodes from $\mathscr{M}$ randomly. With more possible nodes the distribution of messages for each Mix node decreases and the Mix nodes need more time to receive their corresponding k_i messages to be able to flush them. This is an interesting point since increasing the size of $\mathscr{M}$ increases the privacy provided by the system but on the other hand introduces a clear observable penalty.

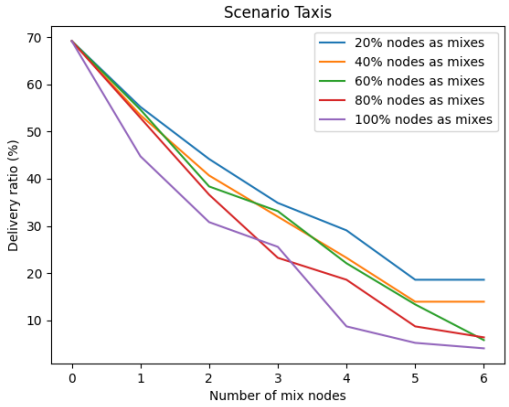
In Figures 4.4 and 4.5 we show the latency of delivered messages for each scenario using different sizes for the set $\mathscr{M}$. Those sizes are 100%, 80%, 40%, and 20% of the total number of nodes for each case. The delay is also shown for different number of



(a) Cambridge



(b) Info5



(c) Taxis

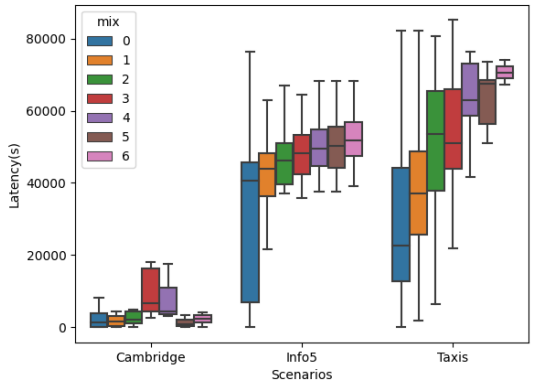
Figure 4.3: Delivery ratio with centrality Mix nodes.

cascaded Mix nodes. As the previous figures, 0 number of Mix nodes is the case of sending messages without the use of the opportunistic Mixnet.

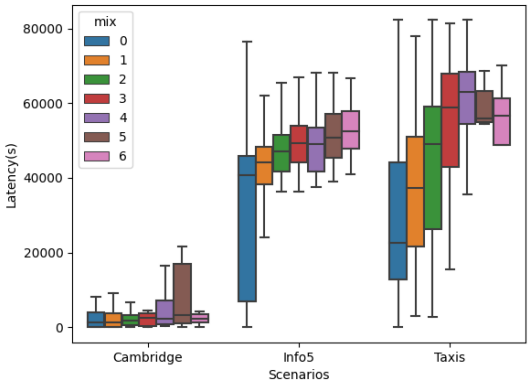
Again, Figure 4.4 shows the case when the set of Mix nodes is chosen randomly and Figure 4.5 when those nodes are selected by their centrality.

The results obtained in this case are quite analogous to the ones observed for the delivery ratio. When using nodes based on their centrality as Mix nodes, we obtain slightly better results. The same happens for smaller sizes of the set $\mathcal{M}$.

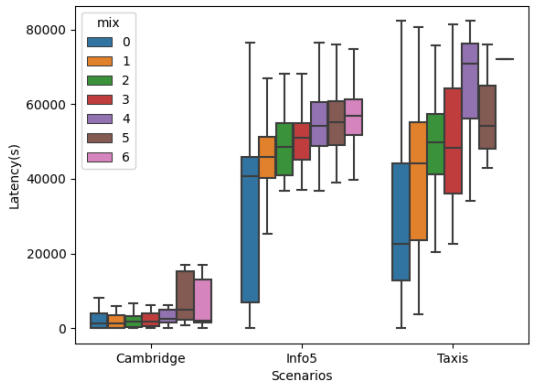
In overall we can see that the penalty introduced by using the opportunistic Mixnet approach is not very large and could be easily tolerable in most scenarios and applications of opportunistic networks.



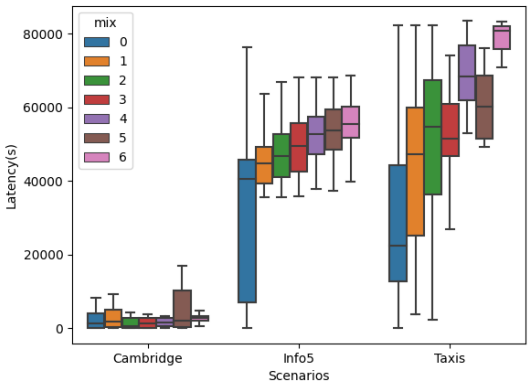
(a) Size of $\mathscr{M}$: 100%



(b) Size of $\mathscr{M}$: 80%



(c) Size of $\mathscr{M}$: 40%



(d) Size of $\mathscr{M}$: 20%

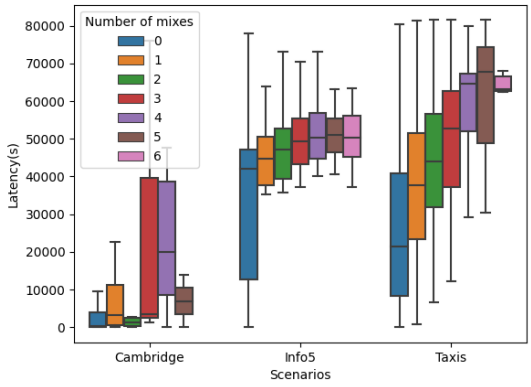
Figure 4.4: Latency with random Mix nodes.

4.6 Discussion

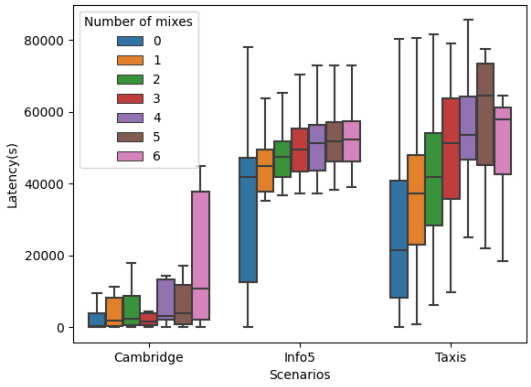
In this section, we present the discussion of the anonymity and performance of Mix networking strategy and show its good performance and privacy.

4.6.1 Protection against traffic attacks

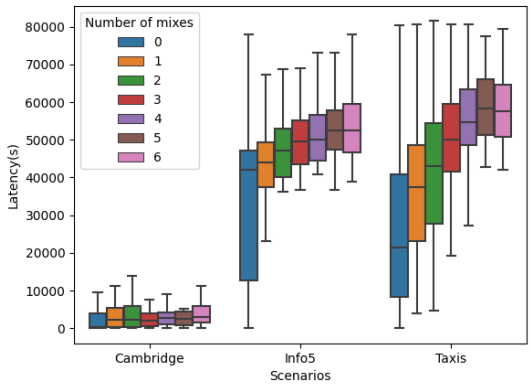
Traffic analysis can be exploited to trace the link between the sender and receiver. Traffic analysis can be divided into two types: passive traffic attacks and active traffic attacks. For passive traffic attacks, an attacker A will monitor the whole system traffic to get the traffic pattern. To resist such kind of attack, Mix networking schema can make each packet forwarding asynchronous and make it hard to predict packet timings. An active



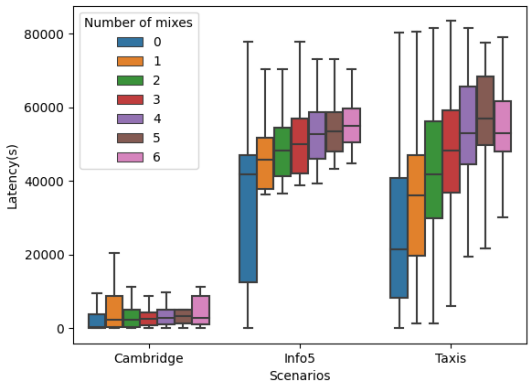
(a) Size of $\mathscr{M}$: 100%



(b) Size of $\mathscr{M}$: 80%



(c) Size of $\mathscr{M}$: 40%



(d) Size of $\mathscr{M}$: 20%

Figure 4.5: Latency with centrality Mix nodes.

traffic analysis attacker can generate some traffic with an specific pattern, which can be used to trace the link between one end of communication path and identify it at the other end. However, as we can choose the range of Mix nodes and the number of Mix nodes on the path, it will be difficult to locate the real communication on both sides. In order to enhance the resistance of traffic analysis, as we will describe in the later part, the system can generate dummy packets. Through these dummy traffic, the disconnected system can be constructed as more dense network. Thus, the attacker would find it more difficult to conduct traffic analysis.

Net	20%	40%	60%	80%	100%
0	3.6	3.6	3.6	3.6	3.6
1	3.5	3.4	3.4	3.7	3.8
2	3.4	3.5	3.5	3.3	3.6
3	3.3	3.5	3.7	3.9	3.5
4	3.2	3.8	3.5	3.5	3.7
5	3.4	3.3	3.4	3.7	3.6
6	3.3	3.5	3.6	3.7	3.7

(a) Cambridge.

Net	20%	40%	60%	80%	100%
0	3.7	3.7	3.7	3.7	3.7
1	3.9	4.0	4.0	3.8	3.8
2	3.9	3.9	4.0	4.1	3.9
3	3.8	4.0	3.9	3.8	4.0
4	3.7	4.0	3.8	3.7	4.0
5	3.7	3.8	3.8	3.9	3.9
6	3.6	3.9	3.7	3.7	3.8

(b) Info5.

Net	20%	40%	60%	80%	100%
0	5.6	5.6	5.6	5.6	5.6
1	5.7	5.4	5.5	5.5	5.6
2	5.8	5.5	5.4	5.6	5.7
3	5.4	5.2	5.5	5.5	5.6
4	5.7	5.4	5.7	5.8	5.7
5	5.9	5.8	5.6	5.5	5.7
6	6.0	5.2	5.1	5.9	6.2

(c) Taxis.

Table 4.2: Average routing step path length with random Mix nodes.

4.6.2 Resistance to compromised Mix nodes and user nodes

In our threat model, the Mix nodes and user nodes can be compromised, but both the entry and exit nodes have to be honest (non-compromised). This means that messages can be transmitted through the compromised nodes provided the exit and entry nodes are not. As we know from our Mix networking structure, each Mix node knows only the previous and next node in a routing path. The entry and exit node know the sender and receiver of the message respectively. Therefore, unless the route only goes through a single node, compromising a Mix node won't trivially enable an attacker to probe the sender-receiver privacy.

4.6.3 Anonymity

Anonymous communication with a Mix network ensures the anonymity of the sender and received through multi-hop Mix nodes. To ensure anonymous communication in OppNets, our proposed method firstly selects nodes as Mixes. Secondly, a user will select a different routing path according to the time. On this level, the attacker cannot detect

Net	20%	40%	60%	80%	100%
0	3.3	3.3	3.3	3.3	3.3
1	3.2	3.7	3.6	3.9	3.7
2	3.2	3.6	3.4	3.6	3.3
3	3.0	3.4	3.6	4.0	4.1
4	2.9	3.7	3.7	3.7	3.4
5	3.1	3.6	3.6	4.0	3.7
6	2.9	3.7	3.6	3.9	4.0

(a) Cambridge.

Net	20%	40%	60%	80%	100%
0	3.9	3.9	3.9	3.9	3.9
1	3.7	3.7	3.9	3.8	3.8
2	3.8	3.8	3.6	3.9	3.8
3	3.4	3.7	3.6	4.0	4.0
4	3.6	3.8	3.6	3.6	3.9
5	3.5	3.7	3.7	3.8	3.9
6	3.3	3.8	3.7	3.9	3.9

(b) Info5.

Net	20%	40%	60%	80%	100%
0	6.2	6.2	6.2	6.2	6.2
1	5.6	5.7	5.7	5.9	5.7
2	5.3	5.5	5.7	5.5	5.7
3	5.3	5.3	5.5	5.6	5.3
4	5.4	5.3	5.5	5.6	5.8
5	5.4	5.3	5.5	5.7	5.5
6	6.4	5.4	5.4	5.4	5.3

(c) Taxis.

Table 4.3: Average routing step path length with centrality Mix nodes.

the correspondence between the sender and receiver in the network directly. The Mix networking strategy can use these two parameters (number of Mix nodes in the network and number of Mix nodes in the path) to make the path selection more random. The attacker cannot get the solid routing model to analyze the behavior of nodes in OppNet. Since the delay tolerant characteristic of OppNet, the encryption and decryption of each message can still tolerate dalys in the data transmission process as one can set a relative long TTL to each message. Thus, Mix networking is a suitable tool to provide anonymous communication in OppNet.

4.6.4 Performance

As from our empirical evaluation result, we can have the conclusion that our proposed method is a lightweight schema. The average routing path from one hop to another changes slightly compared with non Mix networking way. In general OppNets, one message needs to be transmitted through several hops.

4.7 Conclusions

In this chapter, we have presented a Mix networking schema to ensure anonymous communication in OppNets. In our scenarios, one can select the number of Mix nodes to use. This kind of methodology can ensure better anonymity and at the same way add slight penalty on latency. We have tested our proposed method on the ONE simulator and the results shows that using nodes with higher centrality to act as Mix nodes can improve the overall delivery ratio.

There are some further methods that we can be used to enhance the anonymous communication with Mix networking in OppNets.

- **Dummy messages.** The use of dummy messages is a common practice in Mixnet systems. These are messages whose only purpose is to confuse possible traffic analysis attacks and at the same time provide enough network traffic to improve the potential delay caused by the Mix nodes. There are several different types of dummy messages that can be used in our Mix networking schema. The general user node can generate dummy message while there is no message to send during some time. This kind dummy message can be sent to any random receiver in the network or to itself as a loop message. The message can simply be deleted by the receiver or dropped when beyond TTL. Another type of dummy message can be generated by Mix nodes. A Mix node can exploit some independent time series function to generate dummy messages (it means that the probability of generating each packet its equal, so the attack can not get any information from the network traffic analysis). Those dummy message can help the system cover up the message traffic which will misguide the attacker to conduct an active attack like traffic analysis.
- **Mix networking path selection.** The routing path is a critical factor in our Mix networking method. The random routing length increases the uncertainty of the attacker to trace the link between the input message and output message. For each message, the sender can select its routing path according to the number of Mixes and the optional Mix range. However, the dis-connectivity property of OppNets cannot ensure that the message can be transmitted to the destination node. So, selecting a path with more links is helpful.

CHAPTER 5

CONCLUSION AND FUTURE WORK

Opportunistic networking is a promising technology, which can be used to connect network nodes in situations where there might not be a direct path between all nodes. Disruptions and delays are prevalent and contacts between nodes occur in an opportunistic manner. There are plenty of routing protocols proposed to enhance the routing performance in OppNets. Our work focused on studying and providing privacy and anonymous communications in OppNets.

In this thesis, we have defined the problem of anonymous communications in opportunistic networks. We designed the attack model and analyzed the weakness of Opportunistic networks. In our work, we have divided the problem into two research aspects. First we propose the use of an onion routing based mechanism to ensure anonymous communications in predictable opportunistic networks. The second contribution is the proposal to use Mix networking to enhance the anonymous communication in generic opportunistic network.

5.1 Conclusion

Opportunistic networks are potential network structure to connect entities without ensuring the whole connectivity. In this thesis, we study the privacy and anonymity of OppNets. Our work mainly studied from two aspects of OppNets: using onion-based method to ensure anonymous communication in POppNets and using Mix networking to

provide privacy and anonymity for general OppNets.

We studied the predictable opportunistic network and proposed onion routing based schema to ensure anonymous communication in POppNets. We conducted our experiments on a concrete network from the Seattle public bus transportation system. We evaluated path establishment and anonymity degree in this network with variable network density. We have also provided tools to asses the anonymity for the paths. These are concrete anonymity measures that can help in evaluating several characteristics related to path anonymity. It is clear that our approach will be very dependent on the actual network topology and behavior and thus, we provide tools to evaluate the convenience of using onion routing in these types of networks. Our proposal is especially suitable in high density networks, allowing anonymous routing with relatively simple mechanisms as compared to other OppNet solutions. We have used a very simplistic approach for onion routing using only the public keys of the nodes to build the onion layers. We assume that we can directly use each node public key instead of establishing a session key, as this establishment incurs in more penalty than gain. This is convenient in our scenario and in general in OppNets but more advanced solutions could be exploited for other applications.

We have also not considered the performance needed by nodes to perform the cryptographic operations since they are negligible if compared to the network delays. As future work we also consider the combination of this approach with different anonymity techniques. One of such techniques is the introduction of noise or confusion in the network through synthetic network messages. The combination of these two approaches can be useful in for example lower density networks where path predictability could be more difficult to ensure.

Finally, we investigated general OppNets and proposed a Mix networking strategy to support anonymous communication in OppNets. In our schema, each user can be designated as normal node or a Mix node. There are two main parameters that can be selected to set the anonymous communication environment: the Mix nodes range and number of Mix nodes. We implemented the whole schema in the ONE simulator and the result showed the efficiency and possibility to use Mix networking in OppNets.

5.2 Additional Future Work

Besides some future work already outlined in the previous section, we point out the potential research direction to extend our work in the future.

1. Improve anonymity metrics: Anonymous metrics are important measure, and very important to study the anonymity in OppNets. Literature is scarce on such measures and more work can be devoted to evaluate anonymity in such networks.

2. Improve routing efficiency: There are plenty of routing protocols in OppNets. We have relayed in our proposals mainly in generic routing. Source routing for POppNets and epidemic routing for OppNets. However some specific scenarios in OppNets can make use of specific and more advanced routing protocols. As it is relatively common in OppNet routing, anonymity is usually not considered in such protocols.

3. Study the use of dummy traffic to introduce perturbation or confusion: The generation of false traffic can provide anonymity by means of introducing uncertainty to the traffic analysis. This is specially relevant in specific cases where the density of the networks is very low. Although the use of such techniques is relatively common in Mix networking scenarios, given the nature of OppNets it can also be beneficial in the onion routing scheme.

4. Study the impact of using more complex Mix networking strategies: This will help enhance the potential resistance to blending attacks and ensure more sophisticated environment.

BIBLIOGRAPHY

[1] Lloyd Wood, Wesley M Eddy, Will Ivancic, Jim McKim, and Chris Jackson.
Saratoga: a delay-tolerant networking convergence layer with efficient link utilization.
In *2007 International Workshop on Satellite and Space Communications*, pages 168–172. IEEE, 2007.

[2] Thomas Jonson, Jonah Pezeshki, Victor Chao, Kristofer Smith, and James Fazio.
Application of delay tolerant networking (dtn) in airborne networks.
In *MILCOM 2008-2008 IEEE Military Communications Conference*, pages 1–7. IEEE, 2008.

[3] Dimitrios Papakostas, Pavlos Basaras, Dimitrios Katsaros, and Leandros Tassiulas.
Backbone formation in military multi-layer ad hoc networks using complex network concepts.
In *MILCOM 2016-2016 IEEE Military Communications Conference*, pages 842–848. IEEE, 2016.

[4] Kevin Fall.
A delay-tolerant network architecture for challenged internets.
In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 27–34. ACM, 2003.

[5] Aruna Balasubramanian, Brian Levine, and Arun Venkataramani.
Dtn routing as a resource allocation problem.
In *Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 373–384, 2007.

[6] Samo Grasic, Elwyn Davies, Anders Lindgren, and Avri Doria.
The evolution of a dtn routing protocol-prophetv2.

In *Proceedings of the 6th ACM workshop on Challenged networks*, pages 27–30, 2011.

[7] Mary R Schurgot, Cristina Comaniciu, and Katia Jaffres-Runser.
Beyond traditional dtn routing: social networks for opportunistic communication.
IEEE Communications Magazine, 50(7):155–162, 2012.

[8] Giuseppe Araniti, Nikolaos Bezirgiannidis, Edward Birrane, Igor Bisio, Scott Burleigh, Carlo Caini, Marius Feldmann, Mario Marchese, John Segui, and Kiyohisa Suzuki.
Contact graph routing in dtn space networks: overview, enhancements and performance.
IEEE Communications Magazine, 53(3):38–46, 2015.

[9] Kevin Fall and Stephen Farrell.
Dtn: an architectural retrospective.
IEEE Journal on Selected areas in communications, 26(5):828–836, 2008.

[10] Vasco NGJ Soares, Farid Farahmand, and Joel JPC Rodrigues.
A layered architecture for vehicular delay-tolerant networks.
In *2009 IEEE Symposium on Computers and Communications*, pages 122–127. IEEE, 2009.

[11] Nilanjan Banerjee, Mark D Corner, and Brian Neil Levine.
Design and field experimentation of an energy-efficient architecture for dtn throwboxes.
IEEE/ACM Transactions on Networking, 18(2):554–567, 2010.

[12] Mario Marchese, Fabio Patrone, and Marco Cello.
Dtn-based nanosatellite architecture and hot spot selection algorithm for remote areas connection.
IEEE Transactions on Vehicular Technology, 67(1):689–702, 2017.

[13] Zhu Ying, Chao Zhang, Fan Li, and Yu Wang.
Geo-social: routing with location and social metrics in mobile opportunistic networks.
In *2015 IEEE International Conference on Communications (ICC)*, pages 3405–3410. IEEE, 2015.

[14] Said Agoujil, Moha Hajar, Youssef Qaraai, et al.
Optimal cluster head in dtn routing hierarchical topology (drht).
International Journal of Communication Networks and Information Security, 8(2):101, 2016.

[15] Matthew Seligman, Kevin Fall, and Padma Mundur.
Storage routing for dtn congestion control.
Wireless communications and mobile computing, 7(10):1183–1196, 2007.

[16] Yun Li, Ling Zhao, Zhanjun Liu, and Qilie Liu.
N-drop: congestion control strategy under epidemic routing in dtn.
In *Proceedings of the 2009 international conference on wireless communications and mobile computing: Connecting the world wirelessly*, pages 457–460, 2009.

[17] Li Yun, Cheng Xinjian, Liu Qilie, and You Xiaohu.
A novel congestion control strategy in delay tolerant networks.
In *2010 second international conference on future networks*, pages 233–237. IEEE, 2010.

[18] Milena Radenkovic and Andrew Grundy.
Congestion aware forwarding in delay tolerant and social opportunistic networks.
In *2011 Eighth International Conference on Wireless On-Demand Network Systems and Services*, pages 60–67. IEEE, 2011.

[19] Vinícius F.S. Mota, Felipe D. Cunha, Daniel F. Macedo, José M.S. Nogueira, and Antonio A.F. Loureiro.
Protocols, mobility models and tools in opportunistic networks: A survey.
Computer Communications, 48:5–19, July 2014.

[20] K. Scott and S. Burleigh.
Bundle protocol specification.
RFC 5050, IETF, November 2007.

[21] Sushant Jain, Kevin Fall, and Rabin Patra.
Routing in a delay tolerant network.
In *Proc. of SIGCOMM '04*, page 145–158, 2004.

[22] David M. Goldschlag, Michael G. Reed, and Paul F. Syverson.
Hiding routing information.
In *International Workshop on Information Hiding*, pages 137–150, 1996.

[23] Mieso K Denko.
Mobile Opportunistic Networks: Architectures, Protocols and Applications.
CRC Press, 2016.

[24] C Song, Z Qu, and N Blumm.
Limits of predictability in human mobility.
Science, 327(5968):1018–1021, 2010.

[25] David M. Goldschlag, Michael G. Reed, and Paul F. Syverson.
Hiding routing information.
In *Proc. of Information Hiding*, page 137–150, 1996.

[26] Ying Zhu, Bin Xu, Xinghua Shi, and Yu Wang.
A survey of social-based routing in delay tolerant networks: Positive and negative social effects.
IEEE Communications Surveys & Tutorials, 15(1):387–401, 2012.

[27] Wenrui Zhao, Mostafa Ammar, and Ellen Zegura.
A message ferrying approach for data delivery in sparse mobile ad hoc networks.
In *Proceedings of the 5th ACM international symposium on Mobile ad hoc networking and computing*, pages 187–198, 2004.

[28] Wenrui Zhao, Mostafa Ammar, and Ellen Zegura.
Controlling the mobility of multiple data transport ferries in a delay-tolerant network.
In *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, volume 2, pages 1407–1418. IEEE, 2005.

[29] Amin Vahdat, David Becker, et al.
Epidemic routing for partially connected ad hoc networks, 2000.

[30] Thrasyvoulos Spyropoulos, Konstantinos Psounis, and Cauligi S Raghavendra.
Spray and wait: an efficient routing scheme for intermittently connected mobile networks.
In *Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*, pages 252–259. ACM, 2005.

[31] Philo Juang, Hidekazu Oki, Yong Wang, Margaret Martonosi, Li Shiuan Peh, and Daniel Rubenstein.

Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with zebranet.
In *Proceedings of the 10th international conference on Architectural support for programming languages and operating systems*, pages 96–107, 2002.

[32] Anders Lindgren, Avri Doria, and Olov Schelen.
Probabilistic routing in intermittently connected networks.
In *International Workshop on Service Assurance with Partial and Intermittent Resources*, pages 239–254. Springer, 2004.

[33] John Burgess, Brian Gallagher, David D Jensen, Brian Neil Levine, et al.
Maxprop: Routing for vehicle-based disruption-tolerant networks.
In *Infocom*, volume 6, pages 1–11. Barcelona, Spain, 2006.

[34] Cong Liu and Jie Wu.
Scalable routing in delay tolerant networks.
In *Proceedings of the 8th ACM international symposium on Mobile ad hoc networking and computing*, pages 51–60, 2007.

[35] Ling-Jyh Chen, Chen-Hung Yu, Tony Sun, Yung-Chih Chen, and Hao-hua Chu.
A hybrid routing approach for opportunistic networks.
In *Proceedings of the 2006 SIGCOMM workshop on Challenged networks*, pages 213–220, 2006.

[36] David Goldschlag, Michael Reed, and Paul Syverson.
Onion routing.
Communications of the ACM, 42(2):39–41, 1999.

[37] Paul Syverson, Gene Tsudik, Michael Reed, and Carl Landwehr.
Towards an analysis of onion routing security.
In *Designing Privacy Enhancing Technologies*, pages 96–114. Springer, 2001.

[38] Paul Syverson, Roger Dingledine, and Nick Mathewson.
Tor: The secondgeneration onion router.
In *Usenix Security*, pages 303–320, 2004.

[39] Andriy Panchenko and Johannes Renner.
Path selection metrics for performance-improved onion routing.
In *2009 Ninth Annual International Symposium on Applications and the Internet*, pages 114–120. IEEE, 2009.

[40] Robin Snader and Nikita Borisov.
Improving security and performance in the tor network through tunable path selection.
IEEE Transactions on Dependable and Secure Computing, 8(5):728–741, 2010.

[41] Michael Backes, Aniket Kate, Sebastian Meiser, and Esfandiar Mohammadi.
(nothing else) mator (s) monitoring the anonymity of tor's path selection.
In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 513–524, 2014.

[42] Roger Dingledine, Nicholas Hopper, George Kadianakis, and Nick Mathewson.
One fast guard for life (or 9 months).
In *7th Workshop on Hot Topics in Privacy Enhancing Technologies (HotPETs 2014)*, 2014.

[43] Rebekah Overdorf, Mark Juarez, Gunes Acar, Rachel Greenstadt, and Claudia Diaz.
How unique is your. onion? an analysis of the fingerprintability of tor onion services.
In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 2021–2036, 2017.

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

[45] Andrei Serjantov and Richard E Newman.
On the anonymity of timed pool mixes.
In *IFIP International Information Security Conference*, pages 427–434. Springer, 2003.

[46] Aniket Kate, Gregory M Zaverucha, and Urs Hengartner.
Anonymity and security in delay tolerant networks.
In *Security and Privacy in Communications Networks and the Workshops, 2007. SecureComm 2007. Third International Conference on*, pages 504–513. IEEE, 2007.

[47] Cong Shi, Xiapu Luo, Patrick Traynor, Mostafa H Ammar, and Ellen W Zegura.
Arden: Anonymous networking in delay tolerant networks.
Ad Hoc Networks, 10(6):918–930, 2012.

[48] Kazuya Sakai, Min-Te Sun, Wei-Shinn Ku, Jie Wu, and Faisal S Alanazi.
An analysis of onion-based anonymous routing for delay tolerant networks.
In *Distributed Computing Systems (ICDCS), 2016 IEEE 36th International Conference on*, pages 609–618. IEEE, 2016.

[49] G. Vakde, R. Bibikar, Z. Le, and M. Wright.
Enpassant: Anonymous routing for disruption-tolerant networks with applications in assistive environments.
Security and Communication Networks, 4(11):1243–1256, 2011.

[50] X. Lu, P. Hui, D. Towsley, J. Pu, and Z. Xiong.
Anti-localization anonymous routing for delay tolerant network.
Computer Networks, 54(11):1099–1910, 2010.

[51] Vicenç Torra.
Data privacy: foundations, new developments and the big data challenge.
Springer, 2017.

[52] Jelle van den Hooff, David Lazar, Matei Zaharia, and Nickolai Zeldovich.
Vuvuzela: Scalable private messaging resistant to traffic analysis.
In *Proceedings of the 25th Symposium on Operating Systems Principles*, SOSP '15, pages 137–152. ACM, 2015.

[53] Rui Huang, Bidi Ying, and Amiya Nayak.
Protecting location privacy in opportunistic mobile social networks.
In *NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium*, pages 1–8. IEEE, 2018.

[54] Rongxing Lu, Xiaodong Lin, Zhiguo Shi, Bin Cao, and Xuemin Sherman Shen.
Ipad: An incentive and privacy-aware data dissemination scheme in opportunistic networks.
In *2013 Proceedings IEEE INFOCOM*, pages 445–449. IEEE, 2013.

[55] Abdullatif Shikfa, Melek Onen, and Refik Molva.
Privacy in content-based opportunistic networks.
In *2009 International Conference on Advanced Information Networking and Applications Workshops*, pages 832–837. IEEE, 2009.

[56] Xiaojie Wang, Lei Wang, and Zhaolong Ning.

A privacy-reserved approach for message forwarding in opportunistic networks.
In *2017 IEEE 31st International Conference on Advanced Information Networking and Applications (AINA)*, pages 1070–1075. IEEE, 2017.

[57] Sameh Zakhary, Milena Radenkovic, and Abderrahim Benslimane.
Efficient location privacy-aware forwarding in opportunistic mobile networks.
IEEE transactions on vehicular technology, 63(2):893–906, 2013.

[58] Minsu Huang, Siyuan Chen, Li Fan, and Wang Yu.
Topology design in time-evolving delay-tolerant networks with unreliable links.
In *Global Communications Conference*, 2015.

[59] Li Fan, Huang Minsu, yin Zhiyuan, Zhang Chao, and Yu Wang.
Reliable topology design in time-evolving delay-tolerant networks with unreliable links.
IEEE Transactions on Mobile Computing, 14(6):1301–1314, 2015.

[60] J Fraire and J M . Finochietto.
Routing-aware fair contact plan design for predictable delay tolerant networks.
Ad Hoc Networks, 25:303–313, February 2015.

[61] Petter Holme and Jari Saramäki.
Temporal networks.
Physics Reports, 519(3):97–125, October 2012.

[62] Raj Kumar Pan and Jari Saramäki.
Path lengths, correlations, and centrality in temporal networks.
Physical Review E, 84(1):016105(10), July 2011.

[63] Arnaud Casteigts, Paola Flocchini, Walter Quattrociocchi, and Nicola Santoro.
Time-varying graphs and dynamic networks.
International Journal of Parallel, Emergent and Distributed Systems, 27(5):387–408, October 2012.

[64] Vassilis Kostakos.
Temporal graphs.
Physica A: Statistical Mechanics and its Applications, 388(6):1007–1023, March 2009.

[65] B. Bui Xuan, A. Ferreira, and A. Jarry.
Computing shortest, fastest, and foremost journeys in dynamic networks.
International Journal of Foundations of Computer Science, 14(02):267–285, April 2003.

[66] Carlos Borrego, Joan Borrell, and Sergi Robles.
Efficient broadcast in opportunistic networks using optimal stopping theory.
Ad Hoc Networks, 88:5 – 17, 2019.

[67] Deepak Kumar Sharma, Sanjay Kumar Dhurandher, Divyansh Agarwal, and Kunal Arora.
krop: k-means clustering based routing protocol for opportunistic networks.
Journal of Ambient Intelligence and Humanized Computing, 10(4):1289–1306, Apr 2019.

[68] Satya J. Borah, Sanjay K. Dhurandher, Isaac Woungang, Vinesh Kumar, and Leonard Barolli.
A multi-objectives based technique for optimized routing in opportunistic networks.
Journal of Ambient Intelligence and Humanized Computing, 9(3):655–666, Jun 2018.

[69] Pierangela Samarati.
Protecting respondents identities in microdata release.
IEEE transactions on Knowledge and Data Engineering, 13(6):1010–1027, 2001.

[70] Claudia Diaz, Stefaan Seys, Joris Claessens, and Bart Preneel.
Towards measuring anonymity.
In *International Workshop on Privacy Enhancing Technologies*, pages 54–68. Springer, 2002.

[71] Andrei Serjantov and George Danezis.
Towards an information theoretic metric for anonymity.
In *International Workshop on Privacy Enhancing Technologies*, pages 41–53. Springer, 2002.

[72] Sergio Castillo-Pérez and Joaquin Garcia-Alfaro.
Onion routing circuit construction via latency graphs.
Computers & Security, 37:197–214, 2013.

[73] FRANK Rubin.
Enumerating all simple paths in a graph.
IEEE Transactions on Circuits and Systems, 25(8):641–642, 1978.

[74] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam.
l-diversity: Privacy beyond k-anonymity.
ACM Transactions on Knowledge Discovery from Data (TKDD), 1(1):3–es, 2007.

[75] Jorjeta G. Jetcheva, Yih-Chun Hu, Santashil PalChaudhuri, Amit Kumar Saha, and David B. Johnson.
CRAWDAD dataset rice/ad_hoc_city (v. 2003-09-11).
Downloaded from `https://crawdad.org/rice/ad_hoc_city/20030911`, September 2003.

[76] U. Moeller, L. Cottrell, P. Palfrader, and L. Sassaman.
Mixmaster Protocol Version 2.
Internet-Draft draft-sassaman-mixmaster-03, Internet Engineering Task Force, 2004.

[77] C. Gulcu and G. Tsudik.
Mixing e-mail with babel.
In *Proceedings of Internet Society Symposium on Network and Distributed Systems Security*, page 2–16, Feb 1996.

[78] George Danezis.
Mix-networks with restricted routes.
In Roger Dingledine, editor, *Privacy Enhancing Technologies*, Lecture Notes in Computer Science, page 1–17. Springer, 2003.

[79] Carlos Borrego, Marica Amadeo, Antonella Molinaro, and Rutvij H Jhaveri.
Privacy-preserving forwarding using homomorphic encryption for information-centric wireless ad hoc networks.
IEEE Communications Letters, 23(10):1708–1711, 2019.

[80] Ari Keränen, Jörg Ott, and Teemu Kärkkäinen.
The ONE simulator for DTN protocol evaluation.

In *Proceedings of the 2nd international conference on simulation tools and techniques*, page 55. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2009.

[81] Dimitrios-Georgios Akestoridis.
CRAWDAD dataset uoi/haggle (v. 2016-08-28): derived from cambridge/haggle (v. 2009-05-29).
Downloaded from http://crawdad.org/uoi/haggle/20160828/one, 2016.

[82] Jérémie Leguay, Pan Hui, Jon Crowcroft, James Scott, Anders Lindgren, and Timur Friedman.
CRAWDAD dataset upmc/content (v. 2006-11-17).
Downloaded from http://crawdad.org/upmc/content/20061117, 2006.

[83] Lorenzo Bracciale, Marco Bonola, Pierpaolo Loreti, Giuseppe Bianchi, Raul Amici, and Antonello Rabuffi.
CRAWDAD dataset roma/taxi (v. 2014-07-17).
Downloaded from https://crawdad.org/roma/taxi/20140717, 2014.