UCLouvain

École polytechnique de Louvain

Synchronous Mix Networks for Anonymous Web Communication

Author: **John DE WASSEIGE**
Supervisors: **Olivier PEREIRA, François-Xavier STANDAERT**
Readers: **Jean-Charles DELVENNE, Yves-Alexandre DE MONTJOYE**
Academic year 2018–2019
Master [120] in Data Sciences Engineering

Abstract

As the amount of users' private data collected by companies and institutions is increasing, privacy has become an important issue. In particular, data from users' Internet activity often contain sensitive information such as their personal interests, health or financial situation, and are therefore a threat to the users' right to privacy. Furthermore, recent revelations of surveillance programs and of leakages or misuse of users' private data have raised the need and demand for anonymous web communication. As of today, Tor is the only widely deployed anonymity network used for web communications and provides privacy by routing the user's requests through network relays such that the actual web request is made by a computer different to the one used by the user himself. However, it has been shown that Tor is vulnerable to traffic confirmation by an attacker correlating the traffic coming in and going out of the network. Synchronous mix networks are another type of privacy-enhancing networks that provide stronger security than Tor by additionally forcing the relays along the messages path to wait a fixed amount of time before sending them as a batch. Even though synchronous mix networks guarantee stronger anonymity than Tor, their usage so far has been limited due to their increased latency overhead.

In this thesis, the problem of using synchronous mix networks for web communication between users and high traffic websites with small sized requests is analysed. Current research generalizes web traffic and does not take into account how specific types of web traffic might have favourable properties for ensuring anonymity. Our objective is thus to assess whether focusing on high rate and small-sized traffic can render mix networks usable for web communication while providing better anonymity than Tor. Our contribution is fourfold. First, based on the literature of anonymity networks, a definition and metric of anonymity is proposed and realistic adversaries for web communication are also reviewed. Second, metrics describing the robustness and anonymity of mix networks are defined and evaluated according to their potential to improve state of the art systems. Third, a use case of mix networks for a popular search engine is conducted and the practicality of designing such a mix network is discussed, and a tool to scale and design such network optimally depending on available resources is implemented. Fourth, the Panoramix API, which aims to create and operate asynchronous mix networks for secure messaging, is studied and contributions to extend its use to the specific traffic considered and for synchronous networks have been implemented. The resulting analysis of the search engine use case leads to the conclusion that synchronous mix networks are a viable solution to improve anonymity for this type of traffic given realistic resources. In further work, real logs of user queries could be used to make even more accurate anonymity evaluations.

Keywords: Privacy, synchronous mix networks, traffic confirmation, web communication.

Acknowledgements

First of all, I would like to express my sincere gratitude to my supervisors Prof. Olivier Pereira and Prof. François-Xavier Standaert for the time they dedicated to the many meetings we have had. Their open-minded and critical way of thinking made those meetings very insightful and always helped me to move forward with my work.

This project started almost one year ago, when Prof. Yves-Alexandre de Montjoye introduced me to the exciting topic of computational privacy. I am very grateful to him for giving me the opportunity to spend last summer in his research group at the Imperial College London and giving me my first insights to this field of research. The PhD students and interns at the Computational Privacy Group greatly contributed to my experience with the research team. I found it very motivating and rewarding to be among a group of like-minded individuals passionate about privacy threats linked to technological developments. More particularly, I want to thank Florimond Houssiau, who supervised me during my time there and has continued to help me through this process. I have learned a lot from his ability to challenge new and existing ideas and focus on the impact of different concepts.

Many thanks also go to Prof. Jean-Charles Delvenne, who along with Prof. de Montjoye, agreed to be part of the jury of this thesis. I also want to thank my friends Erik, Kristian and Patrick for reviewing this work and providing valuable comments.

Last but not least, I would like to warmly thank my family for their unconditional support. I am grateful to them for the remarkable value they attach to education, which surely accounts a great deal to where I am today.

Contents

List of Acronyms

AES	Advanced Encryption Standard.
DNS	Domain Name Service.
HTTP	Hypertext Transfer Protocol.
HTTPS	Hypertext Transfer Protocol Secure.
IP	Internet Protocol.
IPv4	Internet Protocol Version 4 (32-bit).
IPv6	Internet Protocol Version 6 (128-bit).
ISP	Internet Service Provider.
IXP	Internet Exchange Point.
MAC	Message Authentication Code.
OR	Onion Routing.
RSA	Rivest–Shamir–Adleman.
SSH	Secure Shell.
TCP	Transmission Control Protocol.
TLS	Transport Layer Security.
URL	Uniform Resource Locator.
VPN	Virtual Private Network.
WWW	World Wide Web.

List of Symbols

u_i	User i.
$\mathcal{U}$	Set of users.
$\mathcal{A}_i$	Anonymity set of user i.
sk_a	Secret key for symmetric encryption (lowercase subscript).
$(\mathsf{pk}_A, \mathsf{sk}_A)$	Public and private key pair for asymmetric encryption (uppercase subscript).
$\mathsf{Enc}_k(m)$	The encryption of plaintext m with key k.
$\mathsf{Dec}_k(c)$	The decryption of ciphertext c with key k.
$\{m\}_{\mathsf{pk}_{\mathsf{pk}}}$	RSA encryption of message m with public key pk_A.
$\Pr[E]$	Probability of event E.
$\mathbf{E}[X]$	Expectation of the random variable X.
μ_X	Mean value of the random variable X.
$\mathbf{Var}[X]$	Variance of the random variable X.
σ_X	Standard deviation of the random variable X.
$H(X)$	Entropy of the random variable X.
$\mathrm{MI}(X;Y)$	Mutual information of the random variables X and Y.

Introduction

Today more than ever, the exponential growth of the Internet has made it an incomparable valuable resource to access information. The opportunities the Internet provides to learn and interact with others and participate in creating meaningful content are enormous. Along with its huge potential to catalyze the improvement of society, the Internet leads to a great deal of challenges. Indeed, the digitalization of information, which contributes to the ease at which content is shared efficiently between users, also enhances one's ability to monitor user behavior. The Prism program of the National Security Agency of the United States and Snowden's revelation of governments surveillance are examples of such monitoring capabilities that are huge threats to the privacy of users, with the users often not being aware of such threats [GM13].

Privacy, anonymity and protection against retaliation of our points of view are however essential for democracy. Since its early age, the Internet has known the need for tools guaranteeing the confidentiality, authenticity and integrity of communications and information sharing between users. Being able to stay private when searching for information (through web search engines), reading what is happening around you (through news sites), or even learning (through encyclopedia sites) is of considerable importance due to the enormous amount of information inherent to the collected data [Sol07]. Information like the users' personal interests, their health or financial situations can often be inferred from their search history.

Solutions to overcome these privacy threats are *needed by everyone*, and not only by the ones having "something to hide" [Sol07], for the very reason that personal data is often collected without consent and then used for commercial, juridic and political purposes without caring about the users' privacy [Cam99]. The ideal solution to enhance the users' anonymity would be to have a trusted central authority through which users could have anonymous access to such information. However, such an authority is very unlikely to exist in a near future when governments themselves are threatening their citizen's privacy [Bau+14].

In response to the expansion of data collected about people and the lack of an ideal solution, privacy solutions in the form of *tools* and *regulations* have been developed both by *businesses* — on the server side – wanting to guarantee the privacy of their customers, but also by *users* — on the client side — concerned about their privacy.

On the *server* side, companies have recently claimed to be offering a "completely private experience" and gained in popularity following recent privacy scandals [Lom18]. In the *web search* business, DuckDuckGo and Qwant, for example, are proposing web search services which aim to be similar to Google while not associating search queries to users. These search engines promote objective and non-personalized search results, non-targeted ads as well as no traceability of the users search history. The same is done in the field of *news websites* (e.g. Medium), promoting both non-profiling and non-targeting ads to their users. For competitiveness reasons, these companies often do not make their code publicly available, and even when they do so they cannot prove that they are actually not using a different version. Therefore the only provided guarantee for privacy is based on user experience of not being tracked, on trust regarding their policies, and on auditing from consulting companies, that also need to be trusted of having enough technical knowledge and of being honest when publishing results. Such guarantees based on *trust* have shown to be insufficient. Companies might indeed not follow their policies such as VPNs providers selling users data [Ola18] or Facebook offering a supposedly private VPN [Con19] to its users only to later reveal that the data going through it had been collected. Furthermore, government agencies may also ask for logs or listen to incoming traffic of these companies [Bau+14].

On the *client* side, the most popular tool for anonymously accessing web content is Tor, with more than two millions directly connecting users [LMD10]. Tor is an overlay network which aims to anonymize the traffic of low-latency applications, such as web browsing, by building encrypted circuits between users in such a way that the web sites are not accessed directly by the user but from another node in the network [DMS04]. However, despite its willingness to provide anonymity, Tor is still facing many challenges both considering its performance [DM09; AG16] and its provided privacy. More precisely, if an attacker observes or controls nodes in the network by correlating traffic entering and leaving from these nodes, the attacker is able to infer if some user has been sending requests to some web site [Lev+04; MD05; Bau11]. This kind of attack is known as *traffic confirmation* and is one of the most popular among which Tor is exposed to. Such attack was already put in place during wars when the military listened to encrypted radio signals in order to infer the location of the attacker [CD07]. To prevent such attack, a solution would be to add latency along the path of data packets and send them in batches. However, because the inner design of Tor is made to enhance anonymous *interactive* communications, this kind of solution is not wished by the designers [DM09].

Multiple other solutions for anonymous communication have been proposed and each one exhibits a certain tradeoff of *performance* versus *security* [Das+18]. Globally, these solutions can be separated into two categories. The first one includes Tor and focuses on *interactive* traffic network by trying to keep *low-latency* overheads. The second one includes solutions based on Chaum *mix networks* in which the nodes of the network, called *mixes*, along with routing encrypted messages, also shuffle them in batch and *introduce delays* [Cha81]. These solutions are often described as *high-latency* ones and try to protect

against more global and active adversaries at the cost of longer waiting times.

As it was described above, there is a real need both for users who want to have access to information on the web without being tracked, and for companies that want to deliver this information while providing strong and trustable privacy guarantees to their users. While there is a demand for low-latency systems performing better than Tor, no solution currently stands out from the rest.

In this thesis, we approach the use of *mix networks* for anonymous web communication on websites having *high traffic and small requests sizes*. More precisely, we restrict ourselves to traffic directed towards *cooperative news, search and encyclopedic* high traffic web sites. By focusing on such web sites having high traffic, we explore to what extent the resistance against traffic confirmation of mix networks can be used while trying to keep the latency overhead acceptable for interactive use.

To approach this problem, we first attempt to define anonymity in Chapter 1. This task requires to precisely contextualize the adversary faced since anonymity has not the same meaning for different users. The expected level of anonymity indeed has to be carefully defined when implementing new solutions. By clarifying multiple concepts related to privacy and anonymity, we precisely formulate the privacy we want to achieve and it serves us as a criterion to evaluate the anonymity of the mix networks later on.

Chapter 2 provides a review of the state of the art low-latency anonymous communication networks. Their strengths and weaknesses against adversaries are also described. Special attention is given to Tor — by far the most popular of these networks — in order to use it as baseline for comparison with mixes. Mixes networks, used for high-latency anonymity networks, are reviewed in Chapter 3. The inner workings of the mix unit and the different batching strategies are thoroughly analyzed as well as the parameters that can be chosen when building mix networks.

In Chapter 4 we deal with the first task of quantifying how much anonymity we get when using synchronous mix networks. This is done by first clarifying the assumptions regarding the considered web traffic. We then introduce new definitions to characterize the influence of topology, dummy traffic and latency on the anonymity and robustness of mix networks. Finally we model an attack on mix networks with a given number of corrupted nodes to assess the fraction of users which are kept anonymous, depending on the introduced parameters.

Lastly, Chapter 5 focuses on the second task of evaluating what is needed to deploy a mix network for popular web search engines. A tool has also been implemented for getting the optimal scale and design of mix networks when inputs such as the number of users, available servers and bandwidth are proposed. The same chapter describes the contributions made to the Panoramix API, which has been developed to build mix network for electronic voting systems and instant messaging. Further extensions that could be used in the Panoramix API and in mix networks are then discussed in an attempt to improve the precision and realism of anonymity evaluation.

Chapter 1

Defining Anonymity

Let us consider the following scenario: four users A, B, C and D are speaking through a telephone. For user A, being anonymous could mean that someone accessing the signals could not decrypt any word that he said during the conversation. The priority of user B could be that an eavesdropper would not know to whom he is talking to. Another user C might as well require that an attacker should not even know that he is communicating, or that he ever has been. The last user D could have a totally different opinion and insist that the location from which he emitted the call should not be known to anybody except himself, not worrying about its actual content. These situations clearly demonstrate that defining anonymity is far from trivial and that one needs to be precise about the context and scope of the problem one is trying to solve.

This first chapter focuses on setting the foundation and clarifying the terminology used in the anonymity literature. Since the focus of this thesis is placed on improving *web* privacy, some background to the inner workings of the web are put forward. The cryptographic primitives that are used by the following chapters are also detailed. In addition, the capabilities of the adversaries on the Web are evaluated with the objective of identifying realistic threats. Based on this, a new description *what level of anonymity we consider enough* is proposed. This provides a baseline used by further chapters to assess and compare the protocols capabilities against adversaries. The chapter ends by going through the attacks against which we wish to provide security.

1.1 Privacy terminology

Privacy can be defined as the "ability one user has to control the dissemination of information about himself" [Din00]. Such information can be separated into multiple aspects depending on the considered context which often relates to either the identity of the user (*who*?), some characteristic about himself (*what*?), or some temporal (*when*?) or locational information (*where*?) about something the user did.

Let us now define the terms allowing us to precisely quantify those aspects using the vocabulary introduced by Pfitzmann and Hansen in [PH08] and summarized by Bauer in [Bau11].

Anonymity set The anonymity of someone can be described as the state of being *not identifiable within a set of subjects*, its anonymity set. This notion is introduced by Chaum in [Cha88]. The notion needs to take place in a certain context, for example when considering the receiver of some search query, the anonymity set becomes the set of users which might have received the response. More precisely, for a message m and a set of users $\mathcal{U}$, we write

$$\mathcal{A}_m = \{u_i \in \mathcal{U} | u_i \text{ might have sent/received } m\}.$$

One quickly realizes that the size of this set is of great importance. Sweeney [Swe02] revisits the concept by introducing the notion of the k-anonymity set. To use the previous example, an attacker trying to identify the actual sender u_{real} of m from the users u_i of the anonymity set $\mathcal{A}_m$ should not have more than $1/k$ probability of success

$$\Pr[u_i = u_{\text{real}} | u_i \in \mathcal{A}_m] = \frac{1}{k} \tag{1.1}$$

A user is *identifiable* when one or more users of the anonymity set have a higher probability of being the actual sender or receiver. Note that since the communication to be anonymized happens between two entities, the definition of anonymity can itself be separated into *sender* anonymity and *receiver* anonymity, depending on the goal of the system.

Unlinkability Two or more users of the system are *unlinkable* if an attacker is not able to distinguish sufficiently whether these users are communicating with each other. The definition also applies for unlinkability of messages. Messages entering a system are said to be unlinkable if after being transformed into ciphertexts by some system, no adversary is able to trace back which message corresponds to a given ciphertext.

Unobservability The two previous properties supposed that the adversary knew that some user was *actually communicating* during the time they used the communication system. More restrictively speaking, some user of a system is said to be *unobservable* if an attacker cannot detect whether the user is using the system to communicate or not. For example, the user might be connected to the system and only be sending dummy traffic so that when the user actually sends a real message, an adversary would not be able to tell the difference.

The need for plausible deniability Even when we have defined anonymity metrics, it is not obvious what one needs to *practically achieve* to ensure a certain level of privacy. Taking the anonymity set metric, if the set size is two, it is obvious that in most anonymous

communications context, the users hopes to be hidden in larger set sizes. On the other hand, having a set size larger than ten billion would almost guarantee perfect anonymity. The latter is practically infeasible and one thus needs a number in between which balances privacy and utility. However, many values are possible in between these two extremes. It is here where the idea of *plausible deniability* comes into play.

In the context of anonymous communications, plausible deniability can be understood as one's ability to deny its "relationship" to some communication by providing sufficient facts to prove one's innocence, or by producing a false account consistent with what wished to be denied, It ensures that some user can not be linked with the communication with a high enough level of certainty. Roe [Roe10] proposes to understand plausible deniability as the complementary property of *non-repudiation*, which means that one is able to provide "irrefutable evidence concerning the occurrence or non-occurrence of an event or action".

1.2 How the Web works

This work focuses on trying to improve the anonymity of Internet users, particularly those making queries on dense traffic websites. Because of this, it is important to understand what happens when a user submits an Uniform Resource Locator (URL) into a browser until the moment the webpage successfully loads in front of the user. Based on this, the kind of attacker we consider is someone with the ability to spy on some steps of this process. Please note that we do not consider the person sitting behind you at the local cafe — watching either what you type in or the page you visit — as being a meaningful adversary. The following section will describe the different building blocks that constitute the Internet.

The Internet stands for Interconnected Network of computers. These computers can generally be divided in two categories. The first one are the *clients* which are the typical user devices accessing the web, and the second one are the *web servers*, which are computers storing the web pages you visit. When connected to the Internet, the computers are uniquely identified in the network using an Internet Protocol (IP) address, which are currently 128-bit long addresses with IPv6 and 32-bit with IPv4.

When a URL is typed in the web browser, you browser needs to find the corresponding IP address. For this it queries the Domain Name Service (DNS) which is a distributed database keeping track of the computer's name and their IP. Once the computer knows where to send the query, the different protocol layers will transform the actual message content into data packets that will be sent throughout the network cables. The protocol stack of the Internet is composed of the following four main components:

1. The application protocol layer makes the link between the user interface (the web browser in our case) and the protocol responsible for the transport of the data packets. The Hypertext Transfer Protocol (HTTP) and Hypertext Transfer Protocol

Secure (HTTPS) are example of application protocols and they allow to use the World Wide Web (WWW), which is basically is collection of documents (and resources like images) linked by hyperlinks.

2. The Transmission Control Protocol (TCP) is responsible for dividing the data received from the application protocol into data packets, adding a TCP header to each of them which contains the port number of the application to which the packet has to be sent. This ensures that these packets will be received unmodified and in the same order as they were sent. On the other hand, when the data packets arrive to one's computer, the TCP strips off the header and makes sure the reassembled data is received by the correct application.
3. The IP layer finally has the role of sending and routing the data packets from one computer to another. In the same fashion as the TCP, it adds an IP header, this time containing the IP address of the sending and receiving computers. Unlike TCP, note that this protocol does not check whether a packet did reach its destination, nor that it was modified along the way.

When a data packet is finally transformed by the IP, it will be go from the user's computer to the Internet Service Provider (ISP), through a phone line for example. The ISP router will then get the intended receiver of the packet by analyzing the header and finally sending it to the corresponding location, hoping by multiple routers on its way before making it the final computer.

Having described the inner workings of the Internet, we are now better able to understand the range of potential threats it is exposed to. Since web servers can see the author of some HTTP request, one is better off having someone send requests in your name, if you wish to remain anonymous. In addition, because all the traffic going out of your computer has to pass through the ISPs routers, someone spying on their servers will have access to the history of who sent which packet and at what time. Even if the content of the data packets is not accessible when they are encrypted, the metadata associated with them — their size, time of arrival, destination address — are all useful information for the attacker. Some governments even have programs requiring the ISPs to give them access to the communication patterns [Mas13] for the purpose of surveillance. Moreover, a competent attacker compromising both the web server by getting access to the communication logs, like the NSA did in their Prism program [GM13], and the ISPs to get the traffic metadata, could lead to the re-identification of web pages queried by millions of users.

As explained earlier the main architectural component of the Internet is the TCP/IP protocol suite, which is not designed for anonymous communication. Indeed, the headers of the data packets containing the receiver IP address makes it inherently difficult to hide the fact that two parties are communicating. This design allows the whole routing protocol to be efficient but is not fundamentally made to allow controlling one's own privacy. All solutions designed to hide one's identity rely both on altering the data packet metadata and on modifying the true sender and receiver.

1.3 Cryptographic primitives

In this section we describe briefly the cryptographic primitives that are used extensively in this work. The first requirement of an anonymous communication system that uses non-private communication lines is that the content of the communication itself has to be encrypted. We write $\mathsf{Enc}_k(m)$ the encryption of message m with key k and $\mathsf{Dec}_k(c)$ the decryption of ciphertext c with key k. We thus detail the two main types of encryption: *private* and *public*; as they are both used in low- and high-latency networks.

In symmetric-key encryption, two users A and B possess the same cryptographic key sk, called the private or secret key. User A encrypts a plaintext message m with its private key sk, which we write as $c = \mathsf{Enc}_{\mathsf{sk}}(m)$. User B then can then get the message back from $m = \mathsf{Dec}_{\mathsf{sk}}(c)$. The encryption performed by symmetric-key encryption uses stream ciphers or block ciphers. The Advanced Encryption Standard (AES) is one of the most popular algorithms for symmetric-key encrypting as well as the one used in Tor.

In the context of anonymous communication, if we want a set of users to communicate with other nodes, a solution could be to have a private key pair between each pair of users. However, this leads to $\mathcal{O}(n^2)$ as the number of users n grows which makes it unviable in practice.

In asymmetric-key encryption, both a public key pk and a private key sk are generated. Any user which has been given pk can encrypt its messages, which can then only be decrypted by the user possessing sk. This solves the above mentioned problem because the user now only needs to share its public key to whoever wants to communicate with him. The Rivest–Shamir–Adleman (RSA) encryption scheme is an example of a widely used public-key encryption scheme. It is based on the difficulty of factoring two large prime numbers and is used by Tor.

1.4 Threat model and adversary capabilities

In the section detailing the inner working of the Web, we mentioned where on the Internet adversaries could corrupt users anonymity. In this section we formalize the threat model and the different categories of attackers, as well as evaluate their realistic capabilities.

A *passive* adversary is able to eavesdrop the data exchanged between the nodes of the network. An *active* adversary on top of this also has the ability to influence the traffic by inserting, deleting or modifying data packets on the network. Each of these two categories of attackers can then be *global* or *local*, depending on the proportion of nodes of the network that they can observe. The global one can act on all, or almost all, nodes of the network. Furthermore, nodes in the network can also be *compromised*, meaning that the adversary can see everything happening inside it, and that he can also modify the packets going through them.

When considering plausible threats models, it is often accepted [KEB98] that a powerful adversary is a global passive one, which can observe all the traffic but can only compromise some fraction of nodes but not a large majority (less than 90%). This model is realistic and is often associated to national entities such as governments agencies [Cam99]. Large corporations sometimes also have enormous resources like national ones but they lack the legal authority to observe the network globally (like being granted access to the logs of the ISPs) [Dan04].

1.5 Anonymity metrics and objective

Ideal world Following the example of cryptographic protocols [HL10], a good path to follow when setting objectives for an anonymous communication system is to consider how an *ideal* system would behave.

For an ideal anonymous web communication system, we expect to have a central trusted authority such that when a user makes a query to get a certain webpage, neither the server, nor any eavesdropper in the network should be able to infer its identity — that is, any form of identification (e.g. its computer or its name). This ideal system should also keep attackers from knowing that the user communicated with that website, even though they don't know what was communicated. This linkability feature might seem way less important than the content at first sight, but in case of any leakage, oppression, or event that would lead the attacker to have access to the history logs of the server which sent the page (or an illegal copy of in case the good server would have deleted it), combining those metadata to the actual content that was communicated would destroy any privacy the user had.

We thus say that a web communication system provides full anonymity when the transcripts produced by all nodes participating in the network are indistinguishable from transcripts where the user would not have communicated.

Qualitative anonymity degrees When introducing anonymity metrics in [RR98], Reiter and Rubin propose characterizing anonymity as a spectrum of six anonymity degrees. These degrees are defined following the value of the probability p that a user has sent some message. In decreasing order of anonymity, we have the following degrees

1. Absolute privacy ($p = 0$)
2. Beyond suspicion ($p = 0.2$)
3. Probable innocence ($p = 0.4$)
4. Possible innocence ($p = 0.6$)
5. Exposed ($p = 0.8$)
6. Provably exposed ($p = 1$)

Using these degrees, anonymity is measured as $1 - p$.

This first description of anonymity is convenient in the sense that it gives a qualitative metric regarding the attacker's confidence in identifying a user. However, it lacks the capacity to describe how *indistinguishable* users are within the anonymity set as a whole [Bau11].

Information theoretic approach Danezis proposes to measure anonymity using the effective anonymity set size [Dan04] S_m for some message m, which is defined as the entropy over the probability distribution of sending m

$$S_m = -\sum_{u_i \in \mathcal{U}} p_{u_i} \log_2(p_{u_i})$$

where p_{u_i} is the probability that user u_i is the sender (or receiver depending on the context) of message m. The interpretation behind this measure is an adversary would need S_m bits to identify the true sender or receiver.

However, this metric first fails in describing the anonymity of a system as a whole, since it only focuses on a particular message. Moreover, when an attack is deployed against a system to identify the sender of some message, the attack either fails or succeeds (in which case the entropy is reduced to zero) [Bau11] and nothing in this metric captures this probability of succeeding since it only considers "average" anonymity overs the users.

Withstand a court challenge The previous metrics, even though they allowed to capture to some extent the anonymity of a communication network, failed in their capacity to quantify one's absolute anonymity. Consider the situation in which a government agency wishes to identify the initiator of some Web request. They managed with various attacks to get an anonymity set containing users Alice and Bob, and each of them has the same probability (i.e. 1/2) of being the true sender. From the information theoretics point of view, the anonymity set, even though it is small, is uniformly distributed and thus provides anonymity. Using the anonymity degrees introduced here above [RR98], we lie between possible innocence and probable innocence since $p = 0.5$. However, in this scenario, the government agency would be able to go to a court law and ask the judge for warrants to investigate the computers of the users.

We might then ask ourselves how many users are needed for the judge to refuse such a demand. It seems established in the community that 30 warrants are enough for the demand to be refused. We will therefore refer to this value as 30-anon when later on evaluating the anonymity provided by our solution.

1.6 Attacks on anonymous communication systems

Let us now describe some general attack models. By general we mean that some usual attack schemes can be applied to various types of anonymous communication systems and that the goal and pattern of the attack stays similar. However, note that in most cases attacks are system-specific, since one system tries to improve security in a particular dimension, it will often be making concessions on some other aspects and thus be more vulnerable to new types of attacks.

1.6.1 Brute force search attack

Brute force search attacks, introduced in [Ray01], happen when an attacker follows a certain message sent from a sender to a node in the network, then keeps track of every possible path for this message by keeping a list of the next nodes to which the previous one sent a message to, repeating this until it finally has a list of possible recipients. In the best case, the list contains all possible users of the network and in the worst case the list contains only one element — the true recipient.

In the same paper, Raymond proposes using paths of random length so that the adversary does not know the size of path to follow and thus the number of possible paths is increased. However, since the length of the paths have to be taken from some random distribution, an adversary could have a good chance getting confidence intervals around the average size of paths and thus it only adds a limited amount of security.

1.6.2 Intersection attacks

Intersection attacks use the fact that when a user sends many messages over a short period of time, the same path is used over multiple rounds, since many data packets have to be sent over this period of time. The attack proceeds as follows: when a certain user sends a message, the attacker makes a list of potential recipients (by doing a brute force search attack for example), and in the next rounds as the user sends other messages, the list of recipients is *intersected* with the new possible recipients. When the probability of two users having the same path is very small — in network topologies that have many paths — this attack has been shown [BPS01] to be very efficient and only a few rounds are needed to recover the actual recipient. Long term intersections attacks are considered by Mathewson and Dingledine [MD04] to be a major unsolved problem in many anonymity systems.

To give an intuitive example, suppose that in a first round the paths A–B–C, A–B–D are taken by two messages which could be sent by A and that in the next round the following paths A–B–E, A–B–C are used. Applying the method described here above, if the time frame is sufficiently small for the messages of some user going through A to be on the same path, the adversary can easily infer that the recipient is C.

1.6.3 Traffic confirmation

This type of attack, also known as end-to-end correlation attack, gets more attention than the others because it is the one that Tor is the most vulnerable to. It is also the kind of attack which mix networks show the greatest advantage over Tor.

Suppose user A_1 wants to send a message m_1 to another user B_1, through a network M (which can seen as a black box for the moment) to avoid B_1 knowing the true sender. Its desired anonymity is that the content of m_1 should be hidden. Now suppose that A sends to M the following encrypted text $\mathsf{Enc}_{\mathsf{pk}_M}(\mathbf{B}, \mathsf{Enc}_{\mathsf{pk}_B}(m_1))$, where $\mathbf{B}$ stands for the address of B. Here an eavesdropper listening to the incoming traffic of M can not understand what is sent since the message is encrypted with M public key pk_M. M then decrypts what he received and forwards $\mathsf{Enc}_B(m_1)$ to B. Note that what is forwarded has no meaning for M since it is encrypted with B's public key. Someone listening to M outgoing traffic won't be able to understand the content of the message either. For this purpose the previously introduced cryptographic primitives are sufficient because they are considered secure.

Now we are in the scenario where A_1 sends a message to B_1 and in the same time frame, there is another user A_2 sending a message m_2 to B_2. The senders now desire that, on top of having the actual content of the messages hidden, someone spying on the protocol traffic can not find out who is communicating with who. If M has the behavior of simply decrypting the received message and forwarding it immediately, then a spy at the exit of M can see that A_1 was the first to send its message and thus deduces that A_1 is communicating with B_1 and A_2 with B_2. This is commonly known as *traffic analysis*. Encrypting the messages and passing them through intermediary nodes, which only decrypt and forward directly, is thus not sufficient. An intuitive solution to this problem would be to make the nodes keep the messages for some time before sending them to the user. Those kind of solution will be discussed in the next chapters and is actually what make synchronous mix networks resistant against traffic analysis.

The previous example was given to provide an intuitive understanding of traffic analysis, which is a general term for analyzing the traffic. However, when we consider correlating incoming and outgoing streams, we call this *traffic confirmation*, which a way of doing traffic analysis. In practice, many different techniques exist and they all have the rely on analyzing and finding sorts of correlation in the traffic exchanges between the nodes of the network. We will give popular some attacks in which traffic analysis has been shown successful.

A first inspiration is to analyze the *correlation coefficient* of two traffic streams. The Pearson correlation coefficient of two random variables X and Y is given by

$$\rho_{X,Y} = \frac{\mathbf{E}[(X - \mu_X)(Y - \mu_Y)]}{\sigma_X \sigma_Y}.$$

where μ_X and μ_Y are the mean values of X and Y, σ_X and σ_Y their standard deviation.

When considering web traffic, the information compared can be time stamps at which nodes emitted some traffic, or the amount of traffic that was emitted at each time. For example, we could have n data points $\{x_1, x_2, \ldots, x_n\}$ and $\{y_1, y_2, \ldots, y_n\}$ representing the amount of traffic (expressed in number of packets) measured at times $t_1, t_2, \ldots, t_n$ for users X and Y.

The correlation coefficient can be extended from random variables to data samples by taking estimates of the numerator and denominators and allowing a certain delay d between the two streams as proposed by O'Gorman and Blott leading to the correlation coefficient of equation 1.2.

$$r_{xy}(d) = \frac{\sum_{i=1}^{n}(x_i - \mu_x)(y_{i+d} - \mu_y)}{\sqrt{\sum_{i=1}^{n}(x_i - \mu_x)^2}\sqrt{\sum_{i=1}^{n}(y_{i+d} - \mu_y)^2}} \tag{1.2}$$

where μ_x and μ_y are the average values of x and y.

Figure 1.1 contains an example of how end-to-end traffic correlation can be performed between two signals. Every second, the number of cells coming in an entry node is measured as well as the number of cells going out some other exit node. Correlation between the two signals is then measured as described earlier.

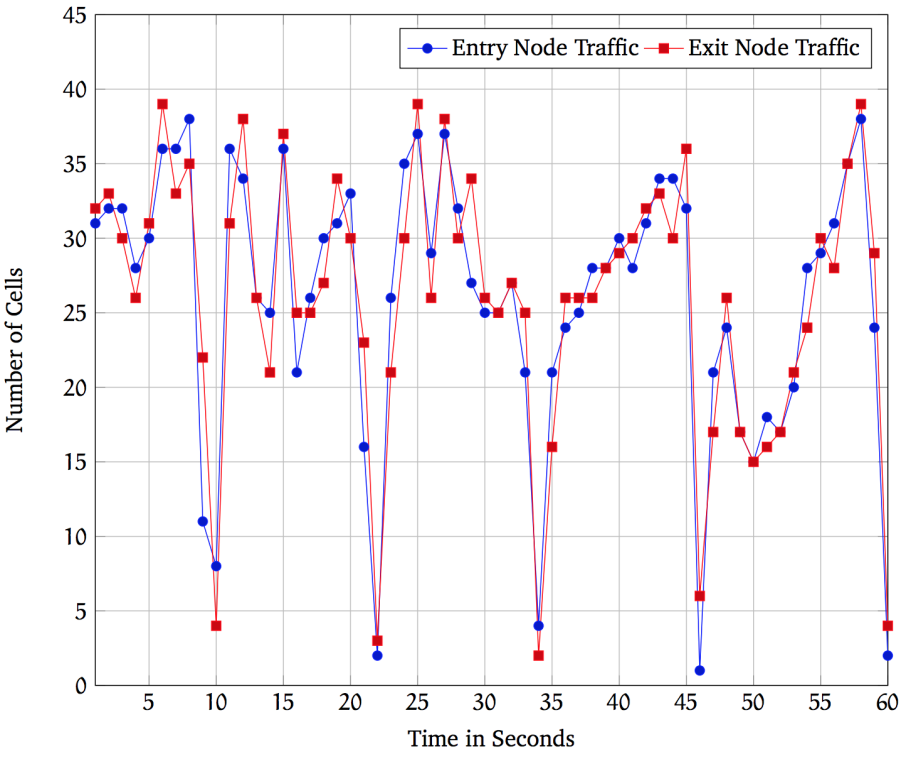


Figure 1.1: End-to-end traffic correlation on Tor traffic [Mül15].

As the correlation coefficient only takes the *linear* relationship into account, the mutual information metric of equation 1.3 can also be used.

$$\mathrm{MI}(X;Y) = \sum_{x \in X} \sum_{y \in Y} p_{(X,Y)}(x,y) \log_2 \left(\frac{p_{(X,Y)}(x,y)}{p_X(x) p_Y(y)} \right) \tag{1.3}$$

where the probabilities are taken by splitting time frames and evaluating the probability of the user to send some packets during this period.

Another traffic confirmation approach proposed in the same paper [OB09] is *packet counting*. Suppose that during some period of time, the traffic going out of some node is equal to the sum of the traffic loads of all the user visited web pages plus any dummy traffic. If an adversary is able to see that a certain number of packets left some server and that approximately the same number of packets went to some user, the latter may well become identified. This is formalized by counting the number of packets x and y of two nodes X and Y and letting

$$\Pr[X \text{ communicates with } Y] \propto \mathsf{dist}(x,y) \qquad \text{where} \quad \mathsf{dist}(x,y) = \sqrt{(x-y)^2}.$$

In a different fashion, Murdoch and Danezis propose a low cost traffic analysis attack against Tor in [MD05]. The attack consists in injecting, from a corrupted node, some traffic of a particular pattern into another Tor node, and observing whether it affects the traffic of some other circuit, in order to know if the communication was passing through this circuit. The correlation is thus made to detect if the injected traffic is correlated with the load of another relay. To quantify this they use a step function $S(t)$ equals to 1 if the corrupted server is sending some packets at sample number t and 0 otherwise, and correlate it with the normalized measured latency $L(t)$ of the target relay at sample t. They measure the correlation c as

$$c = \frac{\sum S(t)L(t)}{\sum S(t)}$$

which can be understood as the correlation between the injected traffic and the latency observed at the target relay.

Chapter 2

Low latency Anonymous Communications Systems

The previous section laid the basis to understand the context of the problem — anonymous communication with a certain type of web traffic — and allowed us to explain why defining anonymity is such a complicated task. Having now a clearer understanding of the existing levels of web privacy one can achieve and of how anonymity can be measured, it is time to see what *solutions* have been proposed and developed by the community.

The development of anonymous communications systems can be traced back to David Chaum's paper "Untraceable electronic mail, return addresses, and digital pseudonyms" of 1981 [Cha81]. Chaum's idea was to hide the correspondence between users by having them send encrypted messages through a path of web servers, i.e. *mixes*, maintaining an asymmetric key pair. These mixes use their private key to decrypt the messages that the users would have encrypted with the server's corresponding public key. After decrypting the messages they would go on delaying and shuffling them before finally sending them either to the next mix — whose address is contained in the message — or to the intended user if the mix was the last of the message's path. Since the introduction of Chaum's 1981 paper, many relay-based anonymity systems have been developed and they are often separated in two categories [DMS04; Bau11; Dan04].

The first category consists of *low latency* solutions. They focus on interactive traffic such as Internet chat, web browsing or Secure Shell (SSH) connections and therefore have a limit on the latency they can introduce in the system, which is usually in the order of a few seconds. This restriction makes it more difficult to prevent traffic confirmation attacks by global attackers, as was detailed in the previous chapter. The second category consists of *high latency* solutions and they have been successful for mailing applications in which the users accept to have a certain delay, as opposed to web communication where short response times are essential. This added latency allows timing information to be hidden and can thus give more protection against traffic confirmation.

This chapter gives a thorough review of the state of the art *low latency* anonymous communications systems that can be used to achieve anonymity for web browsing. Their respective strengths and weaknesses against various adversaries will also be detailed. The next chapter will then present in detail the *high latency* systems based on Chaum mixes, which we just briefly introduced.

2.1 Single hop proxy

The first and simplest anonymous communication system we present is the one in which we have a *single trusted central server* that acts as a web proxy through which the users web requests and responses pass before going to or coming from web servers. The trusted relay is thus in charge of stripping information, which may be a threat to user anonymity, before sending the actual request to the web server.

The objective of single hop proxies is thus to have all its traffic as if it originated from the proxy so that the web server does not receive information pertaining the true sender. On top of this, when the traffic towards the proxy is encrypted, it hides the true destination of the user's traffic against eavesdroppers located between the user and the proxy.

The Anonymizer and SafeWeb are examples of software implementing this idea. As a way to strip sensitive information they to filter out Javascript or Java applets which could be potential threats in case they execute code on the user's computer, as done in [MS02]. Other implementations include BTGuard, IPredator or any other Virtual Private Network (VPN) server. Note that the main difference between a VPN and a proxy is that the former, on top of making your traffic appear as if it came from another IP address, also encrypts the data going from the user computer to the VPN.

However, this type of solution is quite limited in the privacy it proposes for multiple reasons.

- Firstly, a user compromising the proxy itself would be able to uncover the traffic details. Also, this type of solution doesn't protect against traffic confirmation either.
- Indeed, another threat could be some entity capable of eavesdropping on the incoming and outgoing traffic of the proxy, and by correlating traffic could thus uncover the destination of the user's traffic, as it was done in [Sun+02].
- Finally, the entity running the proxy itself could be forced to give away information on what it sees, as it happened with `anon.penet.fi` — an email relay which provided anonymous and pseudonyms email accounts — which was forced to reveal the identity of some of its user because the service was used to show child pornography [Hel96]. The anonymity of such solution therefore highly depends on the integrity of the service and the law to which it is subjected.

2.2 Crowds

Crowds was presented by Reiter and Rubin in [RR98]. Their idea is to provide anonymity in the following way: once a user makes a request, that request is sent in the "crowd" of others users, which they call *Jondos*, and the request bounces between the users until one of them eventually makes the query to the web server. To determine which user is going to actually make the request it uses the following strategy

1. each connected user receives a list of other participants [Dan04] (i.e. the "crowd") and transfer its web request to a randomly selected node in the list;
2. when the next user receives the request, it makes a random experiment such that
   - with probability p_q the user queries the web server with the actual request,
   - with probability $1 - p_q$ he forwards the request to a randomly selected user of the crowd and point **2.** is repeated until the request arrives to the server.

Once the server has gotten the request and sends its reply, the latter is forwarded back to the original user via the path constructed in the sending phase. Figure 2.1 contains an example of such a network of six users making requests to six different servers.

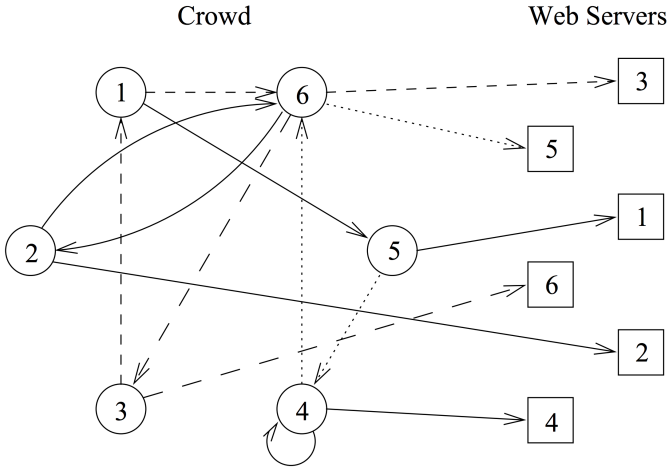


Figure 2.1: Example of a Crowds network architecture with six users sending requests to six Web servers [RR98]. The initiator and server of each path are labeled with the same number. For example, the request initiated by user 6 is first sent to user 3 which then forwards it to the intended web server.

From a security point of view the probability $1 - p_q$ defines the *degree of uncertainty* that some user was the actual sender of the request or not. The goal of Crowds is thus to provide *sender anonymity* but not receiver anonymity. Also note that since the traffic considered in Crowds are HTTP requests, both the actual destination and the content of the data packets are not encrypted.

The main anonymity results they obtain are characterized by *probable innocence*. The path initiator is said to have probable innocence when its probability of being the first collaborator's immediate predecessor, given that a collaborator is on the path, is lower than 1/2. More precisely, given the number of users N, the forwarding probability $p_f = 1 - p_q$ and the number of compromised users collaborating together c, their results states that if equation 2.1 is verified, then the path initiator has probable innocence.

$$\frac{p_f}{p_f - \frac{1}{2}}(c+1) \leq N \tag{2.1}$$

The result implies a balance between the number of collaborating nodes and the length of the path taken by some request, which is proportional to the performance of the system.

Attacks against Crowds have been developed in [Shm02; Wri+02] and they appeared to be efficient in re-identifying the initiator of the requests.

2.3 Tor

Tor was developed by Dingledine, Mathewson and Syverson [DMS04] and stands for The Onion Router. It is based on and improves the Onion Routing (OR) design which has been designed and analyzed in [GRS96; RSG98; SRG00; Syv+01]. The idea behind Tor is that instead of directly exchanging data with the destination, the user data passes through some circuit in forms of packets, using layered encryption so that each node only knows the previous and next hop, and only the exit node is able to see the actual traffic, but not the identity of the initial sender. As it is currently the most popular solution for web anonymity, we will get into more details on its inner workings and existing attacks against it, as there has been a wide variety of these deployed. Explaining those attacks will guide us as a baseline to further assess the security of mix networks.

2.3.1 Network design

The first type of nodes constituting the network are called *onion routers* (or Tor relays) and can be hosted by anyone meeting certain criteria in terms of bandwidth availability. They are in charge of connecting to the requested destination and of relaying the data. Each onion router has a long-term identity key to sign a summary of its keys, address, bandwidth, exit policy, etc. (i.e. the router descriptor) and a short-term onion key to decrypt requests sent by users to set up circuits and exchange new temporary keys. There are two special types of Tor relays

- *Entry guards* are the first nodes of circuits. They are usual relays that require to have been active for a certain amount of time.
- *Exit nodes* are the last hops of circuits. They require a higher bandwidth capability since they will be exchanging the actual traffic with the web servers.

The second type of node is on the user side where a local software called an *onion proxy* (or a Tor client) is run to connect to the Tor network. Its role is to get the onion routers list and related information, to establish circuits across the network, and to manage the connections from the user applications.

The third type of node are called *directory servers*. They are trusted nodes responsible for analyzing the network and relaying information about it to the relays and clients. Once a user connects to the network, it first gets a list of available relays from a directory server before establishing the circuits. Since those servers contain detailed information about the network structure and relays as well as be able to provide information for onion proxies, they must be trusted and available most of the time.

Other type of network nodes include *bridges* and *hidden services*. The first are are relays that are not listed in the directory servers in order to hide some of them from attackers. They are often used when users are under some kind of censorship by the governments forcing the ISPs to block the traffic going to the known Tor relays). Hidden services are anonymous servers which can connect to the Tor network. They were made in order to provide *responder anonymity*, meaning that web servers are able to be accessed without providing their IP address.

2.3.2 Cells and circuits

When a user starts using Tor, the first message he sends allows a circuit to be opened through the network by having a route labelled. The messages are then routed on this predetermined path using a particular label for each of them. The anonymity is provided at the TCP layer where the data is split into equal size packets called Tor cells. Figure 2.2 contains a representation of how a two hops circuit is built in Tor and how the user then exchanges data with a website.

2.3.3 Path selection

The path selection algorithm is responsible for choosing the relays taken in the paths of users. The idea is to associate *selection probabilities* to the relays. Since Tor relays are maintained by volunteers having different bandwidth capacities, not all relays can have the same number of circuits going through them. Tor therefore assigns probabilities to the relays depending on their advertised bandwidth as in equation 2.2.

$$\Pr[\text{relay}_i \in \mathcal{P}_i] = \frac{b_i}{\sum_j b_j} \tag{2.2}$$

where b_i is the perceived bandwidth of relay i measured by the directory servers and weighted by a coefficient depending on the position and flag of the relay in the network (guard or exit node) [RP17].

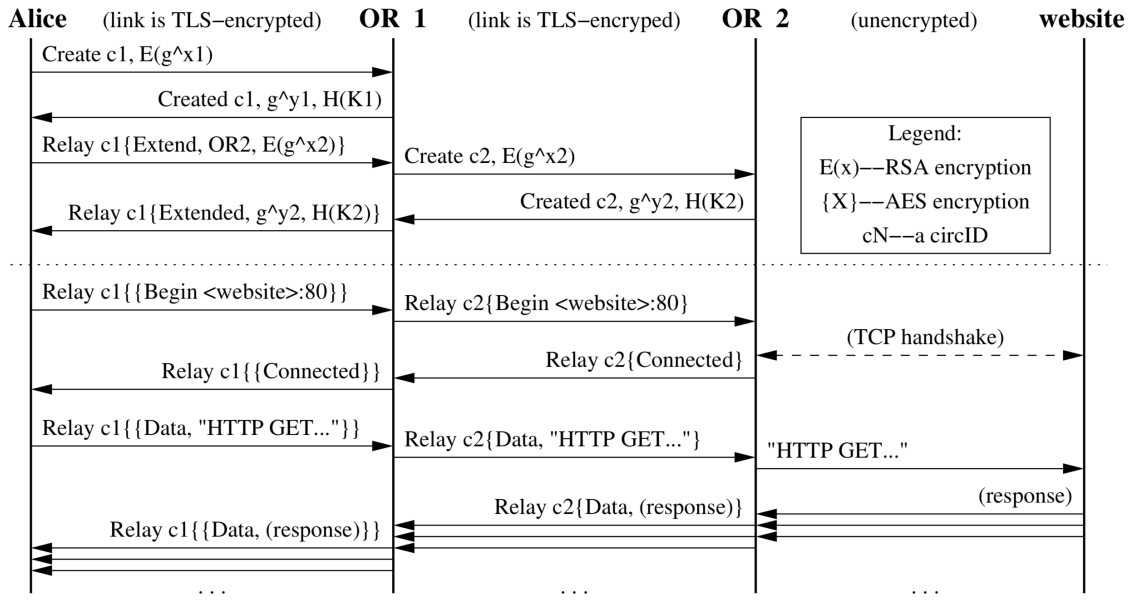


Figure 2.2: Tor circuit (of two hops) construction and data exchange system [DMS04].

2.3.4 Security goals

Tor main security feature comes from the *unpredictability of its routes* and the assumption that an attacker is not powerful enough to observe both the traffic going in the network and coming out of it [Syv11]. However, many people either overestimates Tor's capability in hiding one's identity, either are not aware of all possible vulnerabilities which Tor is subject to.

Tor security goals is to protect against traffic analysis. It considers an "*adversary who can observe some fraction of network traffic; who can generate, modify, delete, or delay traffic; who can operate onion routers of his own; and who can compromise some fraction of the onion routers*" [DMS04]. Tor does not attempt to protect users against a global passive adversary, as this kind of attacker is able to execute end-to-end correlation attacks, which Tor is vulnerable to. Note that Tor does not even implement countermeasures to this attack. In their own words:

> "*Rather than focusing on these traffic confirmation attacks, we aim to prevent traffic analysis attacks, where the adversary uses traffic patterns to learn which points in the network he should attack.*" [DMS04]

2.3.5 Existing attacks

Bandwidth manipulation

As detailed above, Tor's path selection algorithm favors relays proposing greater bandwidth (the bandwidth estimates aren't even often accurate and one can manipulate the perceived bandwidth [Bau11]), an attacker can therefore increase the number of circuits passing through its node by having large bandwidth and maintaining non restrictive exit policies [MW08].

This is not an "identification of users" attack itself, but combined with other attacks, it allows the attacker to control a larger part of the network than what he is supposed to control with its actual resources, by having more traffic pass through his relays than he should have.

Traffic analysis and correlation

As mentioned in the introduction, the main threat for Tor users happens when an adversary is able to observe both the traffic leaving the user computer and going to the entry guard (the first node of the circuit) and the traffic going from the exit node (the last node of the circuit) to the server. The attacker can use traffic correlation by counting the number of packets [SS03] or timing these packets [Lev+04] as detailed in Section 1.6.3.

Many variants and different traffic analysis methods have been put in place and shown to be efficient against Tor [MZ07; Dan05; Cha+14; MD05; Joh+13; BMS01]. This kind of attack becomes very effective when the attacker is able to propose relay having large bandwidth since a lot of circuits then passes through his relay. In this scenario, the security provided by the unpredictability of the multiple-hop circuit looses its strength as having paths going in different countries does not affect what the attacker can observe if there is a significant probability that traffic will pass through his relays.

Network latency and clogging attacks

Two other important types of attacks are worth being mentioned. The *network latency* attacks is based on the fact that Tor circuits are kept the same for a certain time (usually about ten minutes) and that the time it takes for a packet to travel from a certain website to an user will be the same as long as the circuit does not change. One can thus see how long it takes for a packet to reach a destination and observe that the packets taking the same amount of time to reach destination will be linked with the same user (this attack obviously needs the server to cooperate with the entry guard). The other attack is called the *clogging attack* and consist in flooding requests through a path and try to detect the latency spike at the destination.

Website fingerprinting

Finally, website fingerprinting has also been shown to be a threat even when using the Tor network. The main mechanism put in place by Tor to hide packet size information is to have cells of 512 bytes. However, there has been attacks shown to use websites specific related information to re-identify them [Cai+12; WG13]. For example, suppose a page has for its logo an image of a certain resolution, which will always be sent whatever the content of the request response. Because there is no universal metric for the size of images on websites, those can become pretty unique when deep analysis of the webpages is done [Eck10], thereby allowing attackers to identify websites just by looking at the number of packets coming back to the user's computer.

2.3.6 Unbalanced protocol distribution

Figure 2.3 contains the traffic distribution of a Tor relay run for four days in 2007 [Bau11]. We immediately note that the distribution between the protocols is unbalanced. Indeed BitTorrent protocol uses a disproportionately high amount of bandwidth while the number of connections are mainly HTTP requests. Note that Tor tried to use port-based blocking strategies to blocks the TCP file sharing ports but those strategies have been shown to be easy to evade.

This unbalanced distribution is not a problem in itself but when considering adding dummy traffic in the network, the capacity to add such traffic is limited by the bandwidth capabilities of the relays, and it thus raises the concern that for users only making HTTP requests, better guarantees and less congestion could be potentially provided by limiting the traffic to HTTP.

Protocol	Connections	Bytes	Destinations
HTTP	12,160,437 (92.45%)	411 GB (57.97%)	173,701 (46.01%)
SSL	534,666 (4.06%)	11 GB (1.55%)	7,247 (1.91%)
BitTorrent	438,395 (3.33%)	285 GB (40.20%)	194,675 (51.58%)
Instant Messaging	10,506 (0.08%)	735 MB (0.10%)	880 (0.23%)
E-Mail	7,611 (0.06%)	291 MB (0.04%)	389 (0.10%)
FTP	1,338 (0.01%)	792 MB (0.11%)	395 (0.10%)
Telnet	1,045 (0.01%)	110 MB (0.02%)	162 (0.04%)
Total	13,154,115	709 GB	377,449

Figure 2.3: Exit traffic protocol distribution by number of TCP connections, size and number of unique destination hosts. Data collected the traffic on a router for four days in December 2007 with the default exit policy by Bauer in [Bau11].

2.4 Peer-to-peer networks

Another interesting type of solutions are those based on peer-to-peer networks. In these type of networks, every node acts as a relay node and all participants are potential initiators of traffic.

Tarzan is an example of this type of network and was proposed by Freedman and Morris [FM02]. When a user makes a request, a set of nodes is selected to form a route through the network, and messages are sent using encrypted tunnels between these nodes, in the same fashion as Tor. An interesting feature of the first version of Tarzan was that each node only needed to know about a *random subset* of nodes. In the improved version, subsets of nodes are kept but more information is requested about the nodes to get a score of how trusty they are, and the less trusty ones get selected with lower probability. This feature has shown to be effective since it makes it difficult for an adversary to influence the path selection. However, Tarzan has been shown to be vulnerable to some attacks in [Dan04] and are based on the fact that the node selection process is observable by an attacker (since the setup is peer-to-peer) and the attacker uses these communication informations to infer through which users the traffic actually passes.

Another example of such peer-to-peer network is MorphMix introduced by Rennhard and Plattner [RP02], and has globally the same workings as Tarzan except that the path is not determined by the initiator but by the intermediates nodes.

Chapter 3

Mix networks

The previous chapter reviewed existing low-latency anonymous communication systems. An emphasis was made on Tor as it is the most widely deployed one. It was mentioned at the beginning of this previous chapter that anonymous communication systems could be divide in two categories: low-latency and high-latency ones. The building blocks of these high-latency network are mixes, even though it is not mandatory and other relay types could be used, in practice it is always the case. We now focus on reviewing the literature on these high-latency systems.

In this chapter, we start by describing how mixes work as a single unit. We proceed by going through the different parameters that one can choose when they are designed, to then construct a classification of them. We then study how multiple mixes can be combined together to form networks and their different topologies. We then review how they are used as solutions to practical problems, and describe their strengths and weaknesses in their capacity to solve those problems. Finally, we will analyze their level of security with respect to the attacks presented in the first chapter.

To make it easier to understand later in chapter why certain operations are meaningful, let us briefly enumerate a few *practical applications* of mixes. The most important goal of a mix is to be some black box that receives batch of messages, then cryptographically transforms them, and finally outputs them such that no outside observer should be able to guess better than randomly which output message corresponds to which input one. Considering secure electronic voting, when the voters submit their ballots, they have to be provided to the electoral authority in such a way that no ballot should be *linkable* to its original author. Mixes have here an important role to play to ensure that no information about the voter is contained in its output. In e-mail communications, users might want to have anonymity and make sure anybody spying the communication line should not know the receiver of the messages. For this purpose, a server mixing a batch of emails and delaying them would allow the senders to be dissipated in a large pool of plausible senders.

3.1 The mix building block

3.1.1 Inner workings

A mix can be described as a router (i.e. a device which receives and forward data packets between computers in a network) that performs the following set of actions

1. Receive some input data packets $\{a, b, c, d, e\}$ at times $\{T_a, T_b, T_c, T_d, T_e\}$;
2. Perform cryptographic transformations on them to get $\{o_a, o_b, o_c, o_d, o_e\}$;
3. Re-order the packets lexicographically or randomly, i.e. *mix* them;
4. Send the mixed output batch $\{o_b, o_c, o_e, o_a, o_d\}$ at time T_{out}.

Those actions are schematized in figure 3.1.

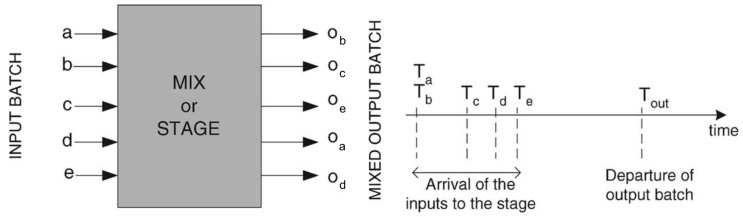


Figure 3.1: Schematic representation of the arrival and departure of mix messages [SP06].

The information contained in the actual content of the input messages is hidden in step *2* when the packets are cryptographically modified and the information provided by the order of arrival is disguised by the mixing and the sending in batch during steps *3* and *4*.

We now describe the exchange of messages happening between the users and mixes and the corresponding encryption and decryption mechanisms.

In order to represent protocol interactions we use the following protocol schema.

Protocol name

Step number) Sender $\longrightarrow$ Receiver (*sending* phase)

Message content

Step number) Receiver $\longleftarrow$ Sender (*response* phase)

Response content

We recall that the encryption of message m with pk_B is written as $\{m\}_{\mathsf{pk}_B}$. However, suppose an adversary sees this ciphertext and tries to guess m by checking if $\{m'\}_{\mathsf{pk}_B} = \{m\}_{\mathsf{pk}_B}$ for many different messages m'. In case the adversary has some other information

regarding the actual content of m (a URL address for example), the set size of possible messages m' could be short enough for the process to be threatened. This problem is overcome by prepending a large random number R to m, which B will discard, and thus sending $\{R, m\}_{\mathsf{pk}_B}$.

Suppose user A wants to send some message m anonymously to server B through mix M and he also expects B to respond him but without knowing A's identity. Let pk_B and pk_M be their respective public keys (note that we suppose B has a known public key but it could also work with symmetric keys by making a key agreement at first).

The *sending* interaction proceeds as follows

Sending through 1-hop mix

1) $A \longrightarrow M$

$$\left\{R_1, \mathbf{B}, \left\{R_0, m, \{A\}_{\mathsf{pk}_M}, \mathsf{pk}_{\mathbf{A}}\right\}_{\mathsf{pk}_B}\right\}_{\mathsf{pk}_M}$$

2) $M \longrightarrow B$

$$\left\{R_0, m, \{A\}_{\mathsf{pk}_M}, \mathsf{pk}_{\mathbf{A}}\right\}_{\mathsf{pk}_B}$$

At this point when B decrypts what he received he gets the following content $m, \{A\}_{\mathsf{pk}_M}, \mathsf{pk}_{\mathbf{A}}$. The two last components of what B received constitutes what is called an *untraceable return address*. It was introduced by Chaum in [Cha81] to allow the receiver of a message to send a response back to the sender without knowing the sender's identity. This is done by adding first an encrypted ciphertext containing A's address, i.e. $\{A\}_{\mathsf{pk}_M}$, such that only M is able to see it, and second a one-time public key for which A has the private key, i.e. pk_A, such that the content of the response of B can not be viewed by M (note that the public key is only used one time otherwise it would mean that B is able to link different messages from A).

Continuing the scenario, the *response* part proceeds as follows

Response through 1-hop mix

4) $M \longleftarrow B$

$$\{R_2, A\}_{\mathsf{pk}_M}, \{R_3, r\}_{\mathsf{pk}_A}$$

5) $A \longleftarrow M$

$$\{r\}_{\mathsf{pk}_A}$$

Notice here that B has no information regarding A's address because it is encrypted with M's public key, and that M can not read the response of B.

A schema summarizing the whole protocol can be found in figure 3.2.

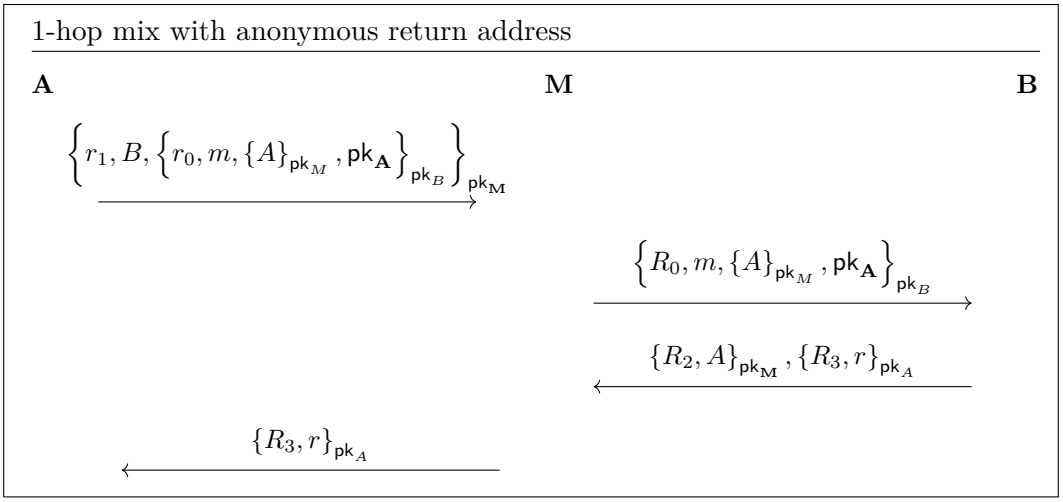


Figure 3.2: Representation of a 1-hop mix with untraceable return address.

3.1.2 Batching strategies

Here we describe the different *batching strategies*, also known as flushing strategies, which might be used for a mix. The batching strategy corresponds to the way in which the input messages, once mixed, are going to be sent out of the mix. Such a strategy can be seen as the algorithm providing answers to the following questions

- What is the condition for a mix to output its batch of messages?
- Does the mix outputs all the messages it received? If not, what proportion does it keep for itself?

The flushing strategies can be divide into two main categories : *simple* and *pool* mixes, which provide an answer to question **2**. The *simple* mixes are the ones firing every request they have received as input when flushing, whereas the *pool* mixes retains a certain amount of messages in the mix. In both categories the mixes can have one or a combination the following characteristics:

1. The *threshold* mix fires when it has collected a certain number N_p of requests.
2. The *timed*, also called *synchronous*, mix fires its batch every t seconds.

This provide an answer to question **1**.

For example, a *simple timed mix* fires all the requests contained in the mix at a certain frequency, whereas a *threshold-and-timed simple mix* fires the requests every t seconds but only if the mix contains at least N_p requests. A more special strategy is one in which the mix fires a fraction of the requests at a certain time, but only when there are more requests than a certain threshold N_p, this is called a *timed-dynamic pool mix*.

Now that we described the different existing types of mixes, we can present a generalized mathematical framework introduced by Diaz and Serjantov in [DS03] to express the batching strategies as arbitrary functions. We start from the observation that a mix can be seen as a function taking as input the number of messages inside the mix and as output the percentage of messages to be flushed. This function is formalized in equation 3.1.

$$\begin{aligned} P : \mathbb{N} &\to [0,1] \\ n &\mapsto P(n) = \{\text{fraction of requests to be fired}\} \end{aligned} \tag{3.1}$$

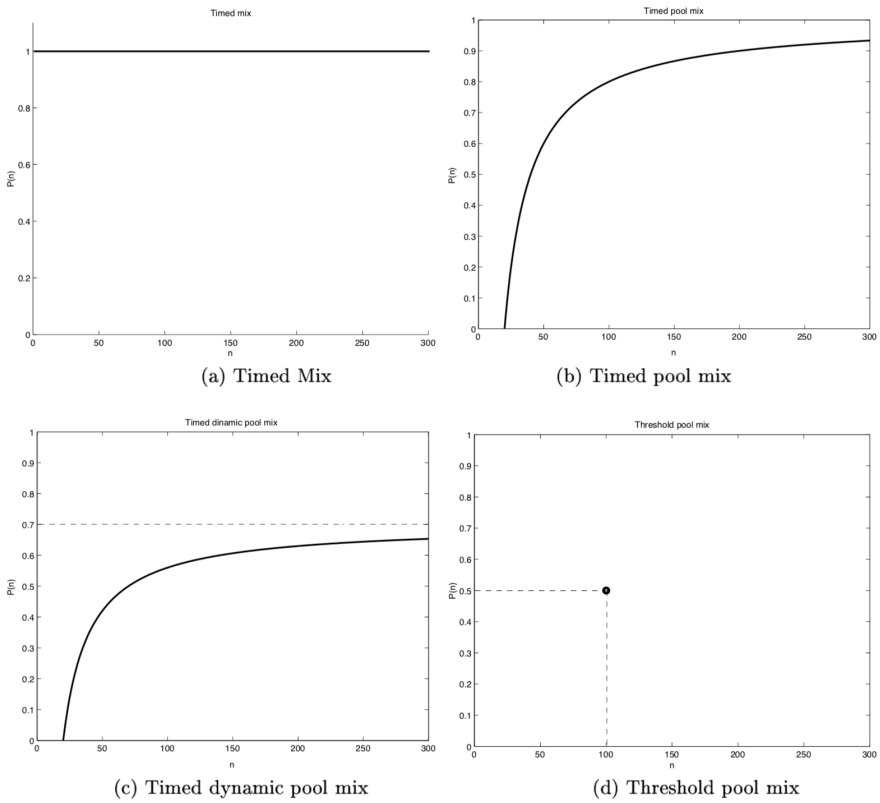
For example, let us build the batching function of the threshold pool mix, with a threshold of N and a pool of N_p messages. The mix flushes $1 - N_p/N$ of its messages at the flush moment which occurs only when n is equal to N, and the rest of time the output of the function should be zero. For this we use the impulse function defined in equation 3.2.

$$\delta_N(n) = \begin{cases} 1 & \text{if} \quad N = n \\ 0 & \text{otherwise.} \end{cases} \tag{3.2}$$

Finally we can write $P_{\text{threshold pool}}(n) = (1 - N_p/N)\delta_N(n)$. Table 3.1 and figure 3.3 contain the flushing functions and associated graphical representations of the different mixes types.

Mix type	**Mix function** $P(n)$
Timed simple mix	1
Timed pool mix	$(1 - N_p/n)$
Threshold simple mix	$1 \cdot \delta_N(n)$
Threshold pool mix	$(1 - N_p/N) \cdot \delta_N(n)$
Timed dynamic pool mix	$c(1 - N_{\min}/n)$

Table 3.1: Types of mixes and their associated flushing function P from equation 3.1. n is the number of messages in the mix. N is the threshold value. N_p is the minimum number of messages which stay in the pool. $\delta_N(n)$ is the impulse function.



(a) Timed Mix (b) Timed pool mix

(c) Timed dynamic pool mix (d) Threshold pool mix

Figure 3.3: Types of batching strategies [DS03]. The y-axis is the fraction of messages $P(n)$ that are fired from the mix and the x-axis the number of messages n in the mix.

3.2 Connecting multiple mixes

Until now we viewed mixes as a single unit and analyzed their inner characteristics. However when designing anonymous communication networks, one need to consider cases where mixes are compromised. The idea is therefore to have each packet pass through multiple mixes, i.e. a *mix network*, to provide stronger robustness against compromised nodes. Packets going through the network will therefore pass through l mixes, and each mix will perform its usual operations.

3.2.1 Extending the cryptographic operations

We first consider the extension of the encryption and decryption procedure of the single mix to the multiple mixes case. When going through l mixes, the idea is the same as with onion routing, where each message will be decrypted once by each mix on its path. The following protocol describes how the sending and response of a message m could be

implemented when passing through mixes M_1 and M_2, i.e. for $l = 2$. For readability reasons, the random strings R_i are implicitly included in the encryptions.

Protocol through 2-hop mix network

1. $A \longrightarrow M_1$

$$\left\{M_2, \left\{B, \{m, \gamma, \mathsf{pk}_A\}_{\mathsf{pk}_B}\right\}_{\mathsf{pk}_{M_2}}\right\}_{\mathsf{pk}_{M_1}}$$

2. $M_1 \longrightarrow M_2$

$$\left\{B, \{m, \gamma, \mathsf{pk}_A\}_{\mathsf{pk}_B}\right\}_{\mathsf{pk}_{M_2}}$$

3. $M_2 \longrightarrow B$

$$\{m, \gamma, \mathsf{pk}_A\}_{\mathsf{pk}_B}$$

4. $M_2 \longleftarrow B$

$$\underbrace{\left\{M_1, \{A, \mathsf{sk}_1\}_{\mathsf{pk}_{M_1}}, \mathsf{sk}_2\right\}_{\mathsf{pk}_{M_2}}}_{\gamma}, \{r\}_{\mathsf{pk}_A}$$

5. $M_1 \longleftarrow M_2$

$$\{A, \mathsf{sk}_1\}_{\mathsf{pk}_{M_1}}, \left\{\{r\}_{\mathsf{pk}_A}\right\}_{\mathsf{sk}_2}$$

6. $A \longleftarrow M_1$

$$\left\{\left\{\{r\}_{\mathsf{pk}_A}\right\}_{\mathsf{sk}_2}\right\}_{\mathsf{sk}_1}$$

3.2.2 Optimizing the cryptographic operations

In the above introduced 2-hop protocol, the public key operations are increasing as the number of intermediary mixes increases. We also observe that the actual size of the content to encrypt varies when more or less mixes are in the message path, or that the content itself of the sent message varies. In response to this increasing complexity in public key operations, multiple solutions have been proposed.

Hybrid mix network This variant uses the public keys of the mixes to encrypt a symmetric key which is then used to encrypt the rest of the message. This leads to the following message being sent from user A to mix M

$$\{\mathsf{sk}_i\}_{\mathsf{pk}_M}, \{\text{rest of message}\}_{\mathsf{sk}_i}$$

The public key operations are here only used to allow less expensive symmetric key operations.

Other implementations of hybrid mix networks have been proposed and they mostly differ in what they encrypt with the public key and how they handle messages of variable length. For example, Goldschlag, Reed, and Syverson propose in [GRS96] to encrypt a key material which then needs to be hashed to get the symmetric key.

ElGamal based mix network Another variant based on ElGamal encryption was proposed in [PIK93] to overcome the fact that the sender needs to encrypt some content in advance *once for every mix* in the path. In their implementation the sender A only needs to send

$$\mathsf{Enc}_{\mathsf{pk}_D}(m) = (g^r, (A, m)\mathsf{pk}_D^r)$$

in which g is the generator, r a random string, and pk_D the general public key of the network obtained from all the public keys of the network mixes [SP06] in the following way

$$\mathsf{pk}_D = \prod_{i=1}^{n} \mathsf{pk}_{M_i} = \prod_{i=1}^{n} g^{d_i} = g^{\sum_{i=1}^{n} d_i}$$

with d_i and pk_{M_i} the private and public key of mix M_i.

The advantage of such implementation is that the encrypted ciphertext only needs one encryption on the user side and is length invariant — its length does not change along its path when being decrypted. However, this kind of solution does not allow for an address of a next mix to be included and therefore the ciphertext has to be sent to all mixes until it is decrypted by every one of them [SP06], which is not a desired property when one wishes to minimize traffic overhead.

3.2.3 Topology

The topology of the network determines how the mixes are interconnected.

We consider three different network topologies as in [DMT10]. For each topology, we explain and represent its design as well as characterize it using the notation introduced by Dingledine, Shmatikov, and Syverson in [DSS05].

For the free route network of n nodes and l layers we write $n \times l$. For the cascade and stratified network we write its dimension as $w \times l$ with w the number of mixes at each layer (the network *width*) and l the number of layers (its *length*). Figure 3.4 contains a schematic represent of the different topologies that are further analyzed. Let us remark that when considering equivalent topology we consider the total number of mixes n to be fixed and thus a 6×2 free route network is compared with a 3×2 cascade and stratified network.

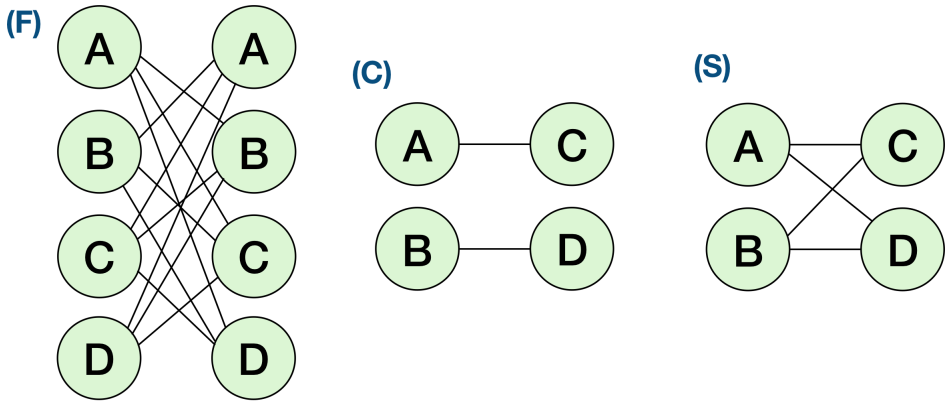


Figure 3.4: Topologies of 4 mixes. *Left*: 4x2 free route (F) network. *Center*: 2x2 cascade (C) network. *Right*: 2x2 stratified (S) network.

3.3 Security properties

There exist many different mixing strategies and each of them comes with its own anonymity, latency and resistance to attacks properties. We here review the security of mixes against usual attacks. Regarding the security of the different batching strategies, it has been studied by Serjantov, Dingledine, and Syverson in [SDS02]. Besides the two attacks that are presented the following subsections, an adversary compromising all the nodes of same path is obviously able to compromise the messages taking those paths.

3.3.1 Intersection attacks

As seen in section 1.6.2, an intersection attack is based on the fact that only a small number of messages will have the same route through the network. In a fully connected network, this kind of attack is very efficient as many different routes are possible [BPS01]. However, when using restricted routes, we are sure that many messages will pass through the same routes. This might bring up the problem of congestion in a network. Since the bandwidth associated with search is relatively small, the traffic relayed by the server allow for a restricted route topology. One may thus wonder in which case designing such a fully connected network is useful then. These questions will be investigated in the next chapter.

3.3.2 $(N_p - 1)$ attack

An $N_p - 1$ consists in active attack flooding a *threshold* mix with $N_p - 1$ messages in order to follow exactly a targeted message. This type of attack is effective with threshold mixes but does not work with synchronous mixes as their batching algorithm is independent of the number of messages that they contain.

3.4 WebMixes

It is worth mentioning that an attempt to build mix networks for Web communication as been done by Berthold, Federrath, and Köpsell [BFK01]. Their implementation is based on a cascade network, where the users have the possibility to choose their path in a list of cascades paths. Dingledine, Shmatikov, and Syverson pointed out some weaknesses of this design in [DSS05]. Even though their implementation is based on mix networks, the mixes do not function in a synchronous way, nor do they offer the possibility to add latency, and are therefore not protected against traffic confirmation.

3.5 Conclusion

In this chapter, we reviewed the inner workings of mixes as well as their batching strategies. The extension from one mix unit to a mix network was then detailed, along with how the different existing topologies. We also analyzed some attacks against certain types of mixes. More precisely, $N_p - 1$ attacks are a type of attacks in which the attackers floods a threshold mix with requests such that the targeted request is isolated and its next node can be identified. The next chapter will focus on analyzing what can be improved to this high latency issue by considering high traffic with small sized requests. Note that as we focus on *synchronous* mixes, $N_p - 1$ attacks are not a concern. However, the problem of high-latency is still a concern for threshold mix networks since the time taken by a request to traverse threshold mixes depends on how fast the other users send messages, and for synchronous mixes, we need guarantees that a sufficient number of messages are sent to avoid having a too small anonymity set.

Chapter 4

Mixes for web search

In the previous chapter, we reviewed how high-latency systems based on mix networks worked. We analysed the mix building blocks and their encryption characteristics as well as their batching strategies, allowing us to seize the potential of synchronous mix networks. Then, we explained what was needed scale from one single mix to a network of mixes. The different parameters involved in the building of such network have also been briefly described, along with the security of mix networks against some attacks.

The objective of this chapter is to answer the first problem we described in the introduction, namely to *evaluate the anonymity we can get by using mix networks with a specific type of web traffic*. This objective can be decomposed into multiple ones and we attempt to achieve them with the following contributions.

Since we consider a specific type of web traffic, we start by precisely describing the assumptions that make this traffic specific and what it implies compared to general traffic. We also study to what extent these assumptions are realistic. Next, since we focus on mix networks, we need to analyse the various parameters which influence the network, such as the added dummy traffic or latency overhead, and quantify how they influence it. Some parameters were introduced in the previous chapter (network topology, for example) but have not been evaluated quantitatively. Then, in order to evaluate the anonymity of mix networks, we need to model an adversary against which the network will have to provide anonymity. We therefore build an adversarial model using the adversary capabilities we introduced in the first chapter, and describe a method to get the optimal choice of nodes the adversary has to corrupt. Finally, we evaluate the anonymity brought by mix networks and the influence on anonymity of the previously studied parameters. In order to evaluate the anonymity, we use the 30-anon anonymity metric introduced in Chapter 1.

4.1 Assumptions

The assumptions we make regarding the specific web traffic considered are the following.

Requests are short. The maximum TCP packet size is 65535 bytes, and for Ethernet a limit is set to 1500 bytes. For the *request* from the user, knowing that the typical headers size are approximatively 800 bytes for applications using cookies, we set a size limit of 1000 bytes, or 1kB. Considering the *response*, Google responses are ±80kB, Google Scholar citations ±1kB, a New York Times news article ±200kB, a Medium article ±120kB, the Wikipedia Philosophy page 386kB and the Wikipedia Differential privacy one is 164kB[1]. Based on these values, we set a threshold for the maximum size to 50kB, corresponding to 50 TCP packets of 1kB. This threshold is realistic as it represents the textual content of the web pages mentioned above. The images and videos are indeed discarded, which seem a reasonable assumption to avoid packet counting. It also makes sense for the network traffic to consider responses 50 times the size of requests.

Requests are stateless. We consider that the attacker is not able to infer information from the previous responses it sent to the user, hence that there is no back and forth. More precisely, suppose a certain web page X is queried by some user, and the only way this user could have requested this webpage, is by getting its URL from another web page Y, that he must have had previously queried. The server will then know that it was the same user which made these two requests. We don't consider this type of exchange, and requests having identifiable *states*, and we suppose that the responses are independent from each other and that the adversary can not use their content to infer new information.

Traffic on the considered websites is high. We expect to have at least a hundred requests every second on the server side, for that the users need to be sufficiently spread out between the mixes, and for it to make sense to have a network of multiple mixes. As a baseline, Table 4.1 contains the traffic rates on popular search, news and encyclopaedic websites.

Web site	**Pageviews per second**
Google	64300
DDG	400
Qwant	30
Startpage	70
Washington Post	90
NYTimes	190
Medium	90
Wikipedia	2800

Table 4.1: Traffic density on popular information sites.

[1]Page sizes obtained using the Network feature of the JavaScript Console in the Chrome browser.

The content of the requests does not identify the user. Suppose some user only makes search queries containing his name or a pseudonym, followed by the actual thing he searches for, or that he only reads news article starting with a precise letter. This kind of behaviour surely compromises the anonymity of the user, as the server itself would be able to see such special pattern. We therefore assume that the content of the requests can not be used to find back its initiator. This assumption is realistic for two reasons. First, no stylometry techniques — which aim to statistically find variations in writing styles between users — have yet been proposed for short sentences. Second, we suppose that the user using a tool to protect his privacy is not going to search for something identifying him. In case we wanted to allow such behaviour, solutions have been proposed to restrict the search query space, and thus provide some security and bounds on the de-anonymization process, even if this largely limits the users potential to search for what they want.

4.2 Evaluation of mix networks parameters

In this section we list and describe the parameters of mix networks. For each of them we analyse their mutual influence and what they imply for the scaling and robustness of networks. The first parameter to be analysed is the topology of the network. Next, we evaluate the influence of dummy traffic and how it scales up depending on the chosen topology and batching strategy. Then, the bandwidth required by the network depending on the real and dummy traffic is analysed. We also describe strategies to spread it among the different nodes, depending on the topology. Finally, we quantify how the latency introduced and the flushing rate influences the number of users and anonymity set, taking into account the number of dummies that were added. This section allows us to gradely built up a global formula which that we use in further sections to quantify the anonymity of mix networks.

4.2.1 Terminology

First we introduce the following terminology. We define the following **variables**

- n the total number of mixes in the network
- n_c the number of compromised mixes in the network
- w the number of mixes per layer
- l the number of layers
- N the number of users per second
- T_s the time taken by the server to process the request and send back the response
- T_m the time it takes for one mix to decrypt and shuffle its messages

- T_f the flushing time of a mix
- T_s the time it takes for one mix to fire requests all its requests
- T_a the added latency in one mix (mix wait while doing nothing)
- $D = N \cdot d_u + n \cdot l \cdot d_m + d_s$ the total number of dummies
- $D_l = D/l$ the average number of dummies per layer
- d_u the number of dummies sent by one user
- d_m the number of dummies sent by a mix
- d_s the number of dummies sent by the server
- b_i the bandwidth of mix i

and the following **constants** values

- $B_q = 1\text{kB}$ the size of one query
- $B_r = 50\text{kB}$ the size of one response
- $L_{\text{mix}} = 5\text{ms}$ the average time to decrypt and shuffle *one* message in a mix when no precomputation is done

4.2.2 Topology

As seen in the previous chapter in Section 3.2.3, there are many different network topologies and they play an important role in the design of the network as they are linked to its scalability and security. Many topology solutions have been studied in the literature and each of them seem to have good arguments for preferring it over another. However, no general quantitative comparison taking both the latency, bandwidth overhead and level of anonymity into account into a globalised framework has been proposed yet. To overcome this lack of global comparison, we attempt to make an it by comparing the different network topologies and their respective strengths and weaknesses.

In order to do this we analyse quantitatively three types topologies given a total number of n mixes. We use the two following *metrics*

1. The number of distinct paths $|\mathcal{P}|$ in the network, which allow among other things to quantify the number of users that are on each path. It is also useful to know how much dummy traffic has to be sent between the different nodes of the network, which we describe in the next subsections.
2. The *entropy* of the network represents how effectively the incoming or outgoing traffic from particular nodes is mixed with traffic from other nodes. It was introduced in [Dan03] by Danezis

$$H(\mathcal{T}) = -\sum_{j=1}^{l}\sum_{i=1}^{w} p_{i,j}\log_2(p_{i,j})$$

where $p_{i,j}$ is the probability that mix i of layer j receives a particular message that was sent from the nodes of the previous layers. For $j = 1$, we use the probability that a node of the first layer is chosen by a user.

Free Route (F) In a free route network all nodes communicates with each other, forming a fully connected network (see left schema of Figure 3.4). The number of paths for a free route network of with n mixes of length l is

$$|\mathcal{P}| = n(n-1)\dots(n-l+1)$$

and

$$p_{i,1} = \frac{1}{n} \qquad p_{i,2} = \frac{1}{n-1} \qquad \dots \qquad p_{i,l} = \frac{1}{n-l+1}$$

Cascade (C) In a cascade network, there are some predefined route and every mix sends all its messages to the next one on the path. (see middle schema of Figure 3.4) The number of paths for a cascade network of length l and width w is

$$|\mathcal{P}| = w$$

and

$$p_{i,1} = \frac{1}{w} \qquad p_{i,2} = 1 \qquad \dots \qquad p_{i,l} = 1$$

Stratified (S) In a stratified network, every mix in one layer can send messages to all mixes in the next layer (see right schema of Figure 3.4). The number of paths for a stratified network of length l and width w is

$$|\mathcal{P}| = w^l$$

and

$$p_{i,1} = \frac{1}{w} \qquad p_{i,2} = \frac{1}{w} \qquad \dots \qquad p_{i,l} = \frac{1}{w}$$

Evaluation We computed the number of paths and entropy values for these topologies and for a total number of mixes $n \in \{6, 8, 10, 12, 14\}$. These values for n were chosen as they sufficiently allow to grasp the behaviour of the metrics considered. The results are for the number of paths and entropy are respectively presented in Figure 4.1 and Figure 4.2. We observe that the free route topology has the most number of paths, as the number of connections in a fully connected graph grows by $\mathcal{O}(n^2)$, while the other are restricted by their smaller width. Regarding the entropy, the stratified network comes first and the free route second.

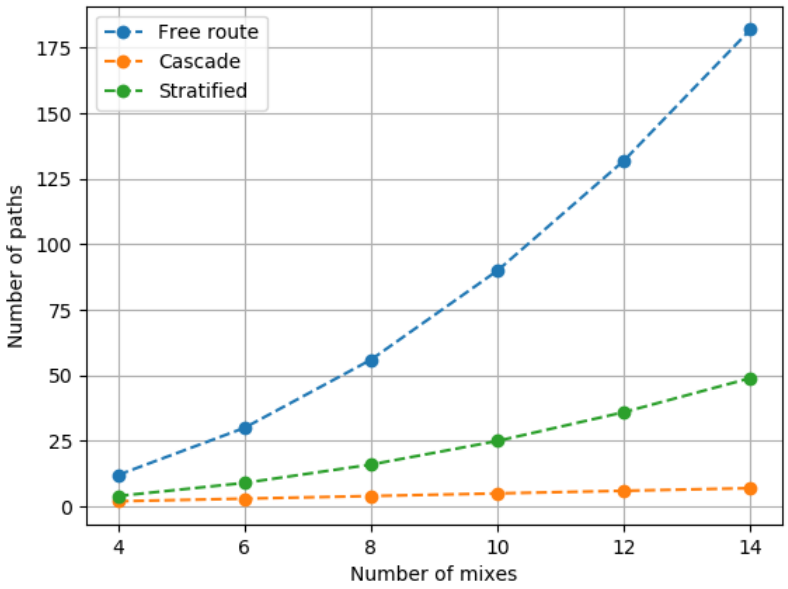


Figure 4.1: Number of paths analysis for different networks topologies of 2 layers.

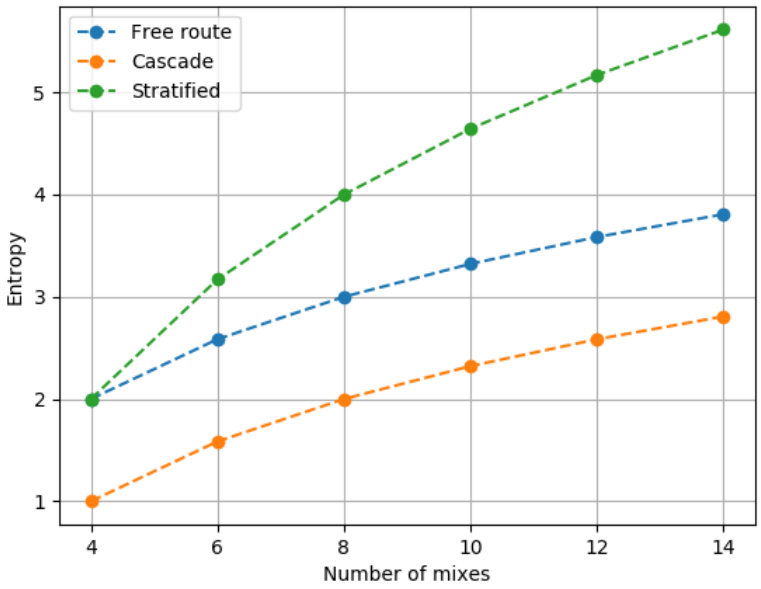


Figure 4.2: Entropy analysis for different networks topologies of 2 layers.

At this point it appears that *free routes* and *stratified* networks are the best choices in terms of number of paths and entropy. However, these metrics do not take into account the actual number of messages that pass through the mixes, whether they are real or dummy, and are therefore not sufficient to represent the anonymity of the users. Indeed, free routes are then more vulnerable to intersection attacks which are based on the fact the probability of packets having the same path is very small, and thus few users will use the same path [Dan03; Dan04]. This number of messages is actually determining the anonymity set size of the users and we thus need to consider the extent to which *dummy traffic* can be added in the network.

Another point to consider is the *bandwidth overhead* these dummies induce in the different topologies. It is known that regarding dummy traffic, free routes network are less scalable [DMT10] since the number of connections between its nodes quickly grows and makes it more demanding in terms of dummy traffic to keep the same anonymity set sizes. On the other hand, when not considering dummy traffic, stratified and cascade networks are said to be less scalable since the same number of queries has to fit in less mixes. The problem is thus to find the right balance.

We thus need to ask ourselves the following questions

- How much dummy traffic is needed to keep a maximum or reasonable number of users per path for the different topologies?
- What bandwidth would this require from the servers?

In the next subsections we try answering these design questions by considering the required dummy traffic and its influence on both the anonymity and robustness of the network.

4.2.3 Dummy traffic

In this section we will analyse the different dummy traffic strategies and evaluate their overhead impact on the different network topologies. Two main dummy traffic strategies are possible: *long-range padding* where each dummy packet sent by the users go through multiple nodes in the same way as real packets, and defense dropping where the dummy packet is only does one hop and is then dropped [Lev+04].

We first quantify precisely the amount of dummies needed to prevent attacks based on the "lack of traffic". Let us recall that despite being *fake*, dummies are indistinguishable from real traffic since they are encrypted as other real messages. Therefore, no mixes along its path can distinguish it from a true message, except for its initiator and the one discarding it (i.e. knowing it is a dummy).

To quantify the amount of dummy needed, we define the two following conditions to be satisfied by the network.

Largest path set Firstly, for the potential number of paths taken by each message to stay large enough, each node of the network needs to make sure it "spreads" enough its messages to different mixes [Ray01]. In order to guarantee a list of recipient of size K, in a network of l layers, each mix should ensure it sends message to at least $\log_l(K)$ different nodes. Note that the sender might also be required to send messages to different mixes to have one additional factor. This requirement gives us a *lower bound* on the number of dummy messages to be introduced in a given network. A *higher bound* is given by the sum of the bandwidth capabilities of the mixes nodes constituting the network minus the required bandwidth to send the real messages. This leads to the following inequalities for D_m the number of dummies sent by each mix

$$\log_l(K) \leq D_m \leq \sum_{i=1}^{n} b_i - N(B_q + B_r).$$

Timing indistinguishability Secondly, to prevent traffic confirmation attacks, the users not only need to send messages to different mixes, but also send multiple messages to the same mix at different time: before, during and after (while it is still connected to the network) the time of sending its real message. The idea is indeed that the traffic sent by user should be the same when he is sending its real message compared to when he is not. However, a good point of using synchronous timed mixes is that dummies needs to be sent only at least once in every mix flushing time. In comparison, low-latency systems often need to send a constant amount of dummy traffic since the relays do not wait before forwarding the packets. This condition can be described with the following inequality for D_u the number of dummies per user

$$\frac{T}{T_f} \leq D_u$$

where T/T_f is the number of times the mix is going to flush in the duration of the user sending its query until he receives the response.

In order to quantify the amount of dummies, a common metric to use [DMT10] is the *dummy overhead factor* o_d defined as

$$o_d = \frac{\text{number of dummies}}{\text{number of real messages}}$$

This overhead factor represents the number of dummies one needs to add in the network for every real message.

The two following conditions are objectives to achieve when introducing dummies in the network.

1. the user sends at least one query for each flushing period T_f of the first mix layer and he has to do this until he gets his response back (the total query-response time is T), T/T_f, where $T = l \cdot T_f + T_s$ (*timing indistinguishability* requirement);

2. the user sends at least one query per mix of the first layer (there are $n_{l,1}$ of them) in each of the mixes during his total activity time in order to keep the number of potential paths large enough (*largest path set* requirement).

This leads to the following lower and upper bounds $\underline{o}_{d,\text{users}}$ and $\bar{o}_{d,\text{users}}$ for the overhead factor of the dummy of users.

$$\underline{o}_{d,\text{users}} = \min\left\{\frac{T}{T_f}, n_{l,1}\right\} \leq D_u \leq \frac{T}{T_f} n_{l,1} = \bar{o}_{d,\text{users}}$$

The chosen upper bound is given by the best case where each user sends something to every $n_{l,1}$ mix in the first layer for every T/T_f batch. This means that every mix receives at each flush a number of messages equal to the number of users, and out of these N messages received, there are $N/n_{l,1}$ which are real ones.

4.2.4 Bandwidth

The bandwidth capabilities of mix i is given by b_i, leading to the following condition on the number of real and dummy traffic incoming the network. Note that the data packets sizes are expressed in bytes while the bandwidth of servers in bits per second, leading to the following bound on N and D_u.

$$8 \cdot ((N + N \cdot D_u) \cdot B_q + N \cdot B_r) \leq \sum_{i=1}^{n} b_i \qquad [\text{bit}\,\text{s}^{-1}]$$

4.2.5 Latency and anonymity set size

A first observation to keep the anonymity set size large enough, is that we need to prevent the adversary to know that the actual message sent was the first one and that there are only four dummies following. So there must be a probability that the user has been sending dummies before sending your actual query, and that the user continues doing so after sending one. Remark that this is totally admissible for search queries since once one is connected there is a short time lapse before the query gets sent and it is in the range of a few seconds.

In the previous section when dummy were introduced, we came with the conclusion that there was N real queries along with $Nw(\frac{T}{T_f} - 1)$ dummy ones. This allow to formulate the following *end-to-end probability of re-identification*

$$\begin{aligned}\Pr[m \in u_i|\text{end-to-end}] &= \max\left\{\frac{1}{\frac{Nw\frac{T}{T_f}}{w}}, \frac{1}{T \cdot N}\right\} \\ &= \max\left\{\frac{T_f}{N(2 \cdot l \cdot T_f + T_s)}, \frac{1}{T \cdot N}\right\}\end{aligned}$$

The maximum is there to take into account the fact that the adversary always has at least a $1/(N \cdot T)$ chance of guessing correctly the user when $N \cdot T$ are plausible.

Let us now focus on the probability for users to be identified when $l-1$ nodes along their path are compromised, in which case the probability is given by 1 over the number of messages per path.

$$\begin{aligned}\Pr[m \in u_i | (l-1) \text{ compromised}] &= \max\left\{\frac{1}{\frac{N\frac{T}{T_f}}{w^l}}, \frac{1}{T \cdot N}\right\} \\ &= \max\left\{\frac{w^l \cdot T_f}{N(2 \cdot l \cdot T_f + T_s)}, \frac{1}{T \cdot N}\right\}\end{aligned}$$

This result confirms what we discussed regarding the networks having a large number of paths. Having a large number of paths indeed makes this probability high and quickly limits the number of plausible users for a given message.

4.3 Adversarial model

In this section we describe the adversarial model that is used to evaluate the anonymity of mix networks in the next section.

Optimal choice of nodes to compromise First, we need to describe this optimal choice implies on the security of the messages passing in the network. In order to quantify the behaviour of network topologies with respect to compromised nodes we define the concept of $(c)_l-\mathsf{compr}$ when a path $\mathcal{P}_i$ of length l has c compromised nodes. To evaluate this we choose multiple scenarios for different topologies in which the adversary could compromise nodes. We decided to use six mix nodes $\{A, B, \ldots, F\}$ since it allows to have enough meaningful dispositions while staying representable on a schema. Figure 4.3 contains the five scenarios we study for the three topologies.

We then evaluate the probabilities that some path on the network is $(c)_l-\mathsf{compr}$ for $c \in \{0, 1, 2\}$ and $l = 2$. While this choice does not encapsulate all the possible values for the number of layers, it allows us to evaluate the evolution of these probability depending on the topology and the choice of compromised nodes. The results are in Table 4.2.

	$(\mathbf{F})$	$(\mathbf{C})_\mathbf{1}$	$(\mathbf{C})_\mathbf{2}$	$(\mathbf{S})_\mathbf{1}$	$(\mathbf{S})_\mathbf{2}$
$\Pr[\mathcal{P}$ is $(0)_2-\mathsf{compr}]$	1/2	0	1/2	0	1/4
$\Pr[\mathcal{P}$ is $(1)_2-\mathsf{compr}]$	1/3	1	0	1	1/2
$\Pr[\mathcal{P}$ is $(2)_2-\mathsf{compr}]$	1/6	0	1/2	0	1/4

Table 4.2: Impact of adversary node control for different topologies. The probability of getting a $(c)_l-\mathsf{compr}$ path in the topologies of figure 4.3 is derived.

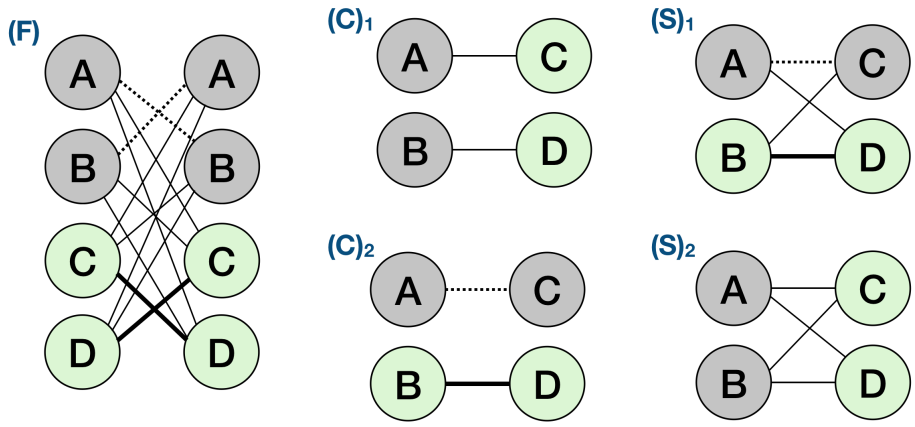


Figure 4.3: Compromised nodes (in grey) possible distribution for different topologies of 4 mixes. (F) is a 4×2 free route. $(C)_1$ and $(C)_2$ are 2×2 cascades networks. $(S)_1$ and $(S)_2$ are 2×2 stratified networks. Non-compromised links are in bold, partially compromised in thin line and fully compromised ones are dotted.

A first observation is that the choice of nodes to compromise in a free route has no impact since a compromised node is appears in every layer. Then, we observe that the best way for a cascade network to compromise nodes is to compromise most of them in a given path. And the same applies for stratified network.

Choosing whether to compromise the nodes of the first layer or the last one depends on what the adversary wants to unveil. By compromising a node in the first layer, the adversary is able to tell *who* is communicating, whereas in the last layer he has more chance to unveil which messages contained *what* content, by collaborating with the server for example.

Robustness The $(c)_l-$compr definition also allow us to hop onto a new metric we introduce to characterise the resistance of a mix network to compromised nodes, which we call *robustness*. The *robustness* r corresponds to the number of non fully compromised paths in a network. Indeed, as we saw previously, once a path of a user does *not* contain a compromised node, it cancels out the adversary's potential since the adversary will loose track of the messages once they pass through the honest mix. More precisely, we define the robustness of some topology $\mathcal{T}$ as

$$r(\mathcal{T}) = 1 - \frac{|\mathcal{P}_c(\mathcal{T}|}{|\mathcal{P}(\mathcal{T}|}$$

with $\mathcal{P}_c(\mathcal{T})$ the set of path containing fully compromised nodes and $\mathcal{P}(\mathcal{T})$ the set of all possible paths for the given topology. Therefore, having a larger number of paths makes it more likely for some path to include at least one non-compromised node.

By considering fractions of compromised nodes of $0\%, 10\%, 20\%, \ldots, 100\%$ on a network of 10 mixes, we obtain the results of Figure 4.4. These results shows us that cascades have a step-like robustness, while free routes and stratified networks have approximately the same behaviour.

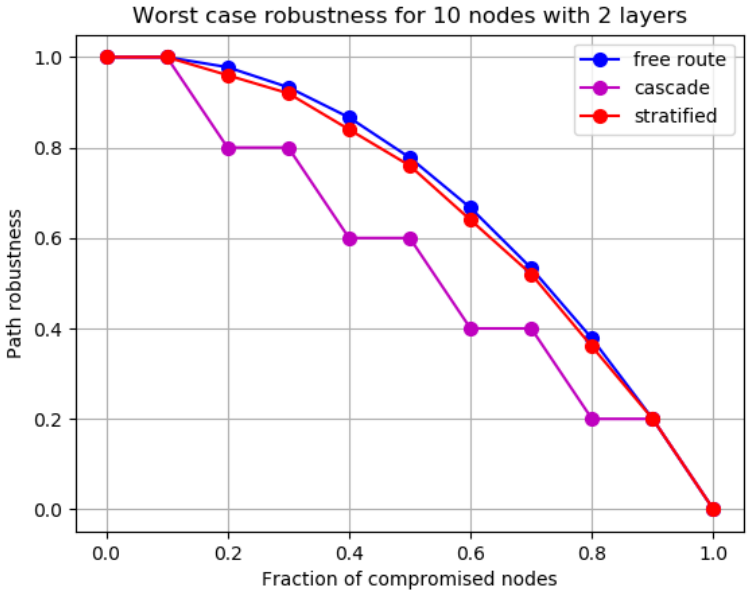


Figure 4.4: Robustness of different topologies.

Attack model Having the optimal choice of compromised node, the last step in building our adversarial model is to describe the attack itself. For this we consider that when no node are compromised at all, all users are secure. For Tor, traffic confirmation comes to play when the adversary controls both the entry guard and the exit node.

As baseline comparison for the Tor attack we use the one introduced in [Syv+01] and repeated in [Wri+02]. The attack thus succeeds when a user chooses one of the $c/2$ compromised nodes of the first and one of the $c/2$ compromised node in the last layer. Since the adversary behaves in an optimal way, it will chooses only the nodes in the first and last layers.

Let us recall that the attack has a single round probability of success given in equation 4.1.

$$\Pr[\text{attacker success on Tor}] = \begin{cases} \frac{c^2}{n^2} & \text{for} \quad l > 2 \\ \frac{c(c-1)}{n^2} & \text{for} \quad l = 2 \end{cases} \tag{4.1}$$

4.4 Anonymity against adversary

This last section is the meeting point of the previous ones. Until now we considered separately every parameter influencing the network and we studied how it contributed to its robustness and anonymity. In this section we evaluate how these parameters behave with respect to the anonymity metric we defined in the first chapter and compare it with Tor. Throughout the section and in the graphs, we note SMix for synchronous mix networks.

Let us quickly recall that our metric $N_{K-\text{anon}}$ gives the absolute number of users such that their anonymity set size is above $K = 30$. The idea behind this was that you would need the set of potential users to be large enough such that the police asking the judge for a mandate for 2 persons would surely be given it but not when 30 are needed. Moreover, having only 2 persons could allow the attacker to combine side information to uniquely identified, which makes it harder the larger the number is. Of course there is no optimal value for K and as we explained it depends on the users and the desired anonymity, but more than 30 allows you to have plausible deniability as well as protection against mandates issued by the court as explained in Chapter1.

Equation 4.2 puts together every property we described while reviewing and evaluating the parameters in this chapter. The intuition behind is that we can have at most N users since there are N users making queries, that the number of layers decreases the number of compromised users as it adds the possibility of having a non-compromised node in one's path, but that there is a limit to this number of layers that depends on the total time and the flushing time, which is itself bounded by the time needed to decrypt and mix the messages.

$$N_{K-\text{anon}} = \min\left\{N, \max\left\{(N+D)\frac{2\cdot l\cdot\beta + T_s}{\beta}\left(1 - \prod_{i=1}^{\alpha}\frac{c_i}{w}\right) - K, 0\right\}\right\} \tag{4.2}$$

where

- $\beta = \max\{T_f, C\cdot L_m\}$ is the maximum between the flushing time and the lower bound on the time taken by the relay to mix the message when the flushing time is too small;
- $\alpha = \min\{l, T/T_f\}$ is the number of layers, or the number of batches possible if the number of layers is too large, leading to no more time to mix and decrypt the messages;
- $c_i \in [0, w]$ is the number of nodes compromised in layer i.

The first results in Figure 4.5 shows the evaluation of the attack for a number of compromised nodes varying between 0% to 100%. The number of 30-anon users is shown and compared for mixes of 3 and 4 layers against the (fictional) Tor with uniform path selection. It appears that the number of layers does not give much additional resistance except when the number of compromised nodes becomes higher than 60%.

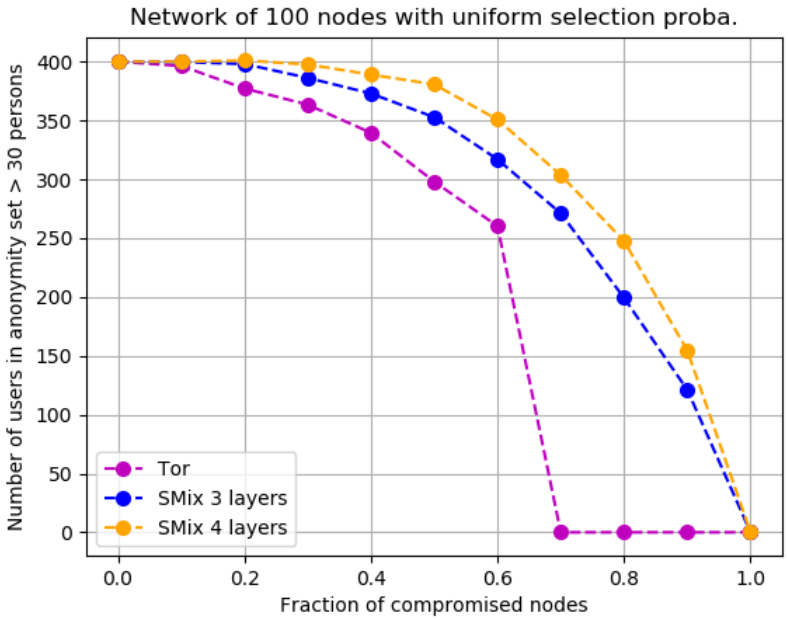


Figure 4.5: Number of users with anonymity set size greater than 30 with varying latency for Tor and synchronous mix networks of 100 relays and a *uniform* selection probability. The comparison is taken for the same total latency.

4.4.1 Latency results

The next results, evaluated for a fixed number of compromised nodes, analyses the influence of our main security parameter: the latency overhead. Figure 4.6 represents the number of users being 30-anon for a network with 80% compromised nodes, 400 users and 200 dummies per second.

4.4.2 Number of layers and dummy traffic results

For three values, 6, 8 and 10 seconds, of fixed latency in Figure 4.6, we evaluate the influence of the number of layers and of dummy traffic.

Heat maps in Figures 4.7, 4.8 and 4.9 contains these evaluations. To visualise easily, for a latency of 6s, on the 200 dummies vertical line, we find back the values of Figure 4.6.

These heat maps allow us to understand the influence of the studied parameters and to know the optimal number of layers and dummies one should send to get the corresponding number of user with 30-anon.

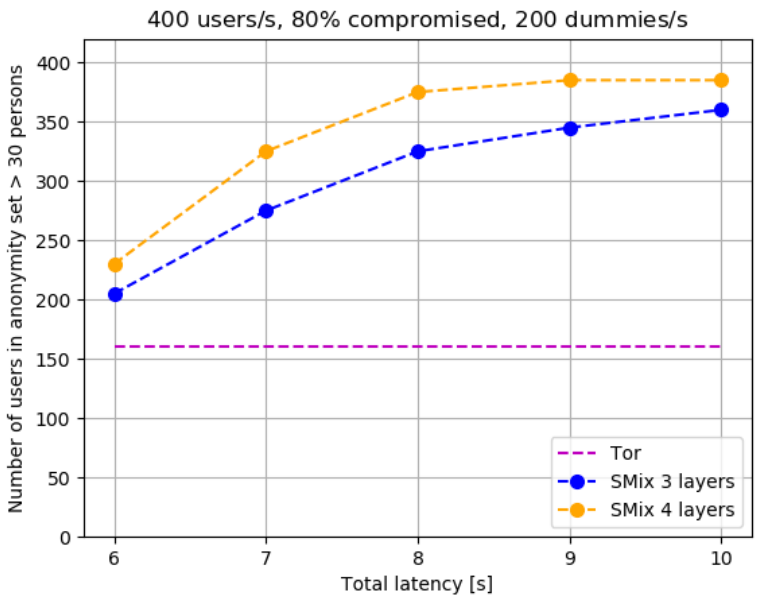


Figure 4.6: Number of users with anonymity set size greater than 30 with varying latency. Based on a model of the Tor network and a synchronous mix networks of 100 relays of which 80% are compromised with 400 requests per second and 200 dummies per second.

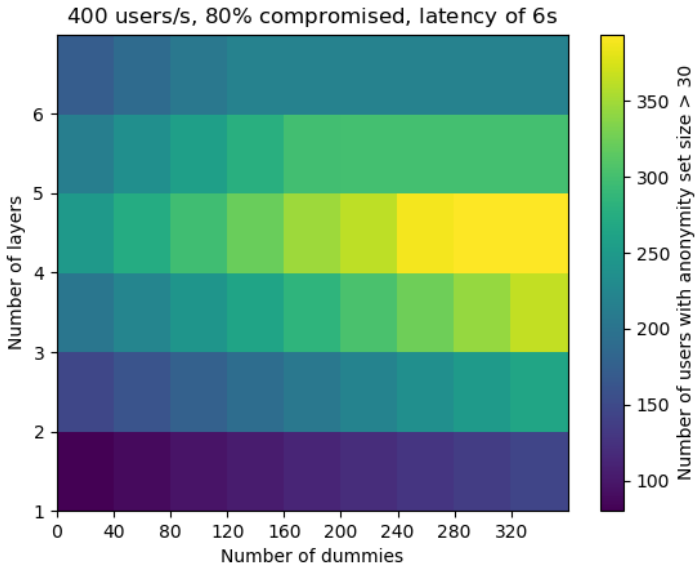


Figure 4.7: Heat map of users with anonymity set size greater than 30 with varying number of layers and number of dummies. Based on a synchronous mix network of 100 relays with 80% compromised, 400 requests per second and a latency of 6s.

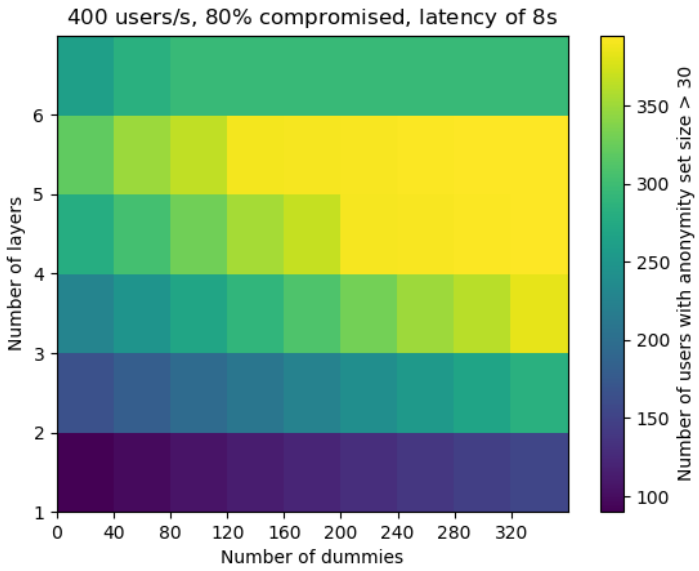


Figure 4.8: Heat map of users with anonymity set size greater than 30 with varying number of layers and number of dummies. Based on a synchronous mix network of 100 relays with 80% compromised, 400 requests per second and a latency of 8s.

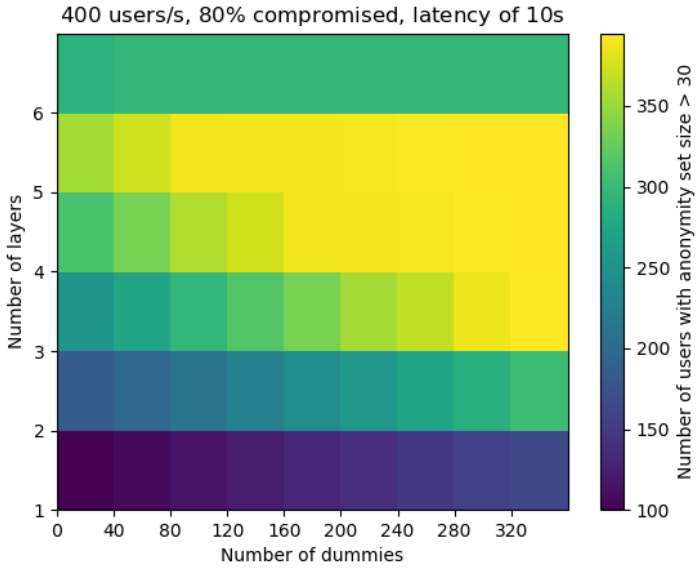


Figure 4.9: Heat map of users with anonymity set size greater than 30 with varying number of layers and number of dummies. Based on a synchronous mix network of 100 relays with 80% compromised, 400 requests per second and a latency of 10s.

Chapter 5

Implementation and scalability

In the previous chapter we analyzed the anonymity provided by mix networks for the particular type of traffic we consider. The influence of the topology, dummy traffic and latency have been evaluated and tested against some adversarial model.

The contribution of this last chapter is threefold.

Firstly, we study what would be needed in terms of network design and capabilities if we wanted to make a real implementation of a synchronous mix network for some known search web site. To do this we analyze the bandwidth distribution of Tor relays in order to know what bandwidth we could expect to have from voluntary servers. Based on this distribution, we evaluate the difference in anonymity that it would imply using the metrics and adversarial model of previous chapter.

Second, we describe a simple tool we implemented which allow to get information on how to design a mix network based on some input parameters such as the query rate and the number of available servers.

Finally, we explain our contribution to an existing API implemented by a European funded research group combining researchers of eleven universities. The API aims to provide functionality to create and operate mix networks, with a focus on mix networks for secure electronic voting and we tried to extend it for web communication. However, there is still some work left for further contributors in order to make it fully usable.

5.1 Realistic requirements and anonymity

In order to design and scale a mix network, we need to choose the node selection probabilities strategy. This section will detail our analysis of the influence of the different node selection probability choices.

5.1.1 Tor bandwidth distribution

To motivate our choice, we first need to analyze what *bandwidth capabilities* one can expect from the relay nodes, which is done in the following subsection. Analyzing these bandwidth capabilities will allow us later on to discuss the impact of having uniform or weighted selection probabilities for the path selection. In the *uniform* case, each relay has a $1/w$ probability of being chosen, where w is the number of mixes in the concerned layer and thus each of them will on average get the same amount of traffic. However, in reality it happens that not all relays have the same bandwidth and that certain nodes will allow a lot of traffic to pass through them while others will only a have small amount available, as already mentioned in Chapter 2 when describing Tor path selection algorithm. In this case, the path selection probabilities are *weighted* and some nodes have a higher probability of being chosen as next hop in a mix path. This method is the one used in Tor where the probability of a relay being chosen is proportional to its advertised bandwidth.

In order to get a representation of what we can expect as bandwidth availability and distribution between the different mixes, we will analyze the Tor network, which is the largest anonymity network as of today. We recall that Tor relays, also called Onion Routers, need to have a minimum bandwidth of $10\text{Mbit}\,\text{s}^{-1}$. An exit relay is considered to be fast when having at least $100\text{Mbit}\,\text{s}^{-1}$. We will thus characterize the traffic on the Tor network by splitting the relays in the following four categories:

1. very fast Onion Router (vf-OR): at least $500\text{Mbit}\,\text{s}^{-1}$
2. fast Onion Router (f-OR): from $200\text{Mbit}\,\text{s}^{-1}$ to $500\text{Mbit}\,\text{s}^{-1}$
3. normal Onion Router (n-OR): from $50\text{Mbit}\,\text{s}^{-1}$ to $200\text{Mbit}\,\text{s}^{-1}$
4. slow Onion Router (s-OR): less than $50\text{Mbit}\,\text{s}^{-1}$

We use Tor Metrics [LMD10] data to get the advertised bandwidth distribution of the relay nodes, which we summarize in Table 5.1. Based on this data, we compute the distribution for the four types of relays we defined. This allow us to get the distribution found in Figure 5.1. It appears that the 65 largest relays, out of 6500 in total, account for 9% of the advertised bandwidth, whereas the 35% of smallest relays only account for 6% of it.

Percentile	100	99	95	90	80	75	60	50	40	30	25
Bandwidth[Mbps]	1700	490	350	250	175	160	110	86	59	34	27

Table 5.1: Advertised bandwidth distribution in $\text{Mbit}\,\text{s}^{-1}$. Each percentile represents the advertised bandwidth that a given percentage of relays does not exceed.

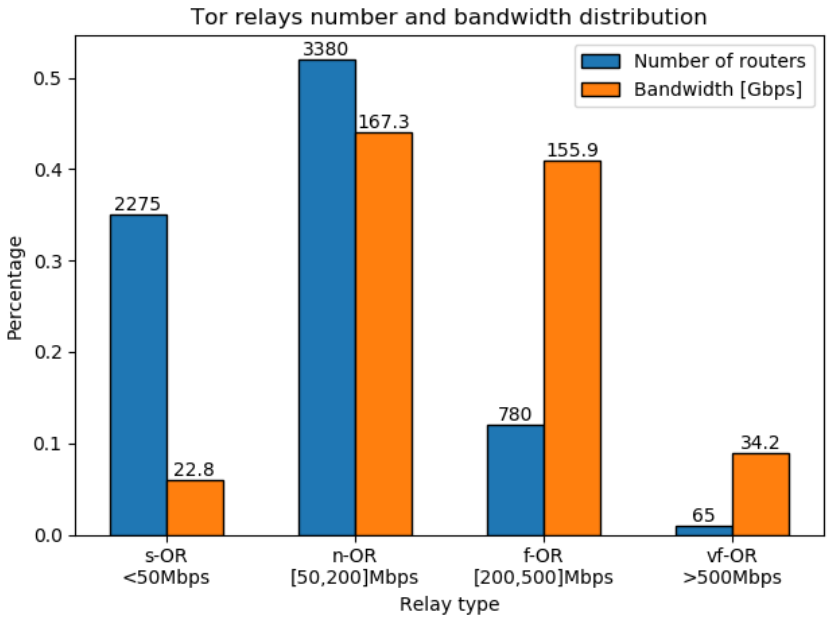


Figure 5.1: Tor number of relays and bandwidth distribution, from Tor Metrics [LMD10] after dividing them into four categories. There are 6500 relay nodes for a total advertised bandwidth of 380Gbit s^{-1}.

Getting back to the goal of this section, we wanted to quantify how the weights of the node selection probabilities were distributed. Let us first clarify some notations to improve readability.

- $P_{\text{Tor},i}$: the probability of a relay of type i to be selected in the weighted coefficient case as in Tor (eg. for a slow Onion Router we have $P_{\text{Tor},s-OR}$)
- $U_{\text{Tor}} = 1/n_{\text{Tor}}$: the probability of a relay being selected in the uniform coefficient case where n_{Tor} is the number of relays in the Tor network
- $C_{\text{Tor},i} = P_{r,i} \cdot n_{\text{Tor}}$: the factor by which the relay of type i is more likely to be chosen as node than it would have in the uniform case

Note that those are fixed coefficients (and thus capitalized) since they represent an observation of the Tor network. We will use them as guidelines when considering weighted probabilities in the node selection process. Combining the advertised bandwidth distribution of Figure 5.1 and equation (2.2), we get the desired coefficients of each relay category in Table 5.2.

	vf-OR	**f-OR**	**n-OR**	**s-OR**
% of quantity	0.01	0.12	0.52	0.35
% of bandwidth	0.09	0.41	0.44	0.06
$P_{\text{Tor},i}$	1.38×10^{-3}	5.26×10^{-4}	1.30×10^{-4}	2.64×10^{-5}
U_{Tor}	1.54×10^{-4}	1.54×10^{-4}	1.54×10^{-4}	1.54×10^{-4}
$C_{\text{Tor},i}$	9	3.42	0.85	0.17

Table 5.2: Summary of Tor relays types number and bandwidth proportion.

The results we obtained allow us to compute the probabilities of different relay types to be selected in the path when considering any mix network structure. Indeed, suppose we want to analyze the influence of uniform versus weighted coefficients of a mix with w nodes in each layer, we can now get the required probabilities

$$\Pr[\text{mix}_i \in \mathcal{P}_{\text{uniform}}] = \frac{1}{w} \qquad \Pr[\text{mix}_i \in \mathcal{P}_{\text{weighted}}] = \frac{w_{\text{Tor}}}{w} P_{\text{Tor},i}$$

where $i \in \{s-OR, n-OR, f-OR, vf-OR\}$.

5.1.2 Influence on anonymity

Using the distribution computed in the previous subsection, we evaluate the anonymity of Tor compared to what Tor would obtain if all relays had the same bandwidth. We put it in comparison with a synchronous mix of 3 layers with similar bandwidth distribution as Tor. The anonymity metric is the same as in the previous chapter. The result are shown in figure 5.2. By compromising 10% of the largest relays, our attack succeeded in compromising 37.5% of the users, while it only compromised around 1% of users in the (fictional) case of uniform distributed relays. We also note that the synchronous mix network perform better than Tor, as they need more relays to be compromised before re-identify users.

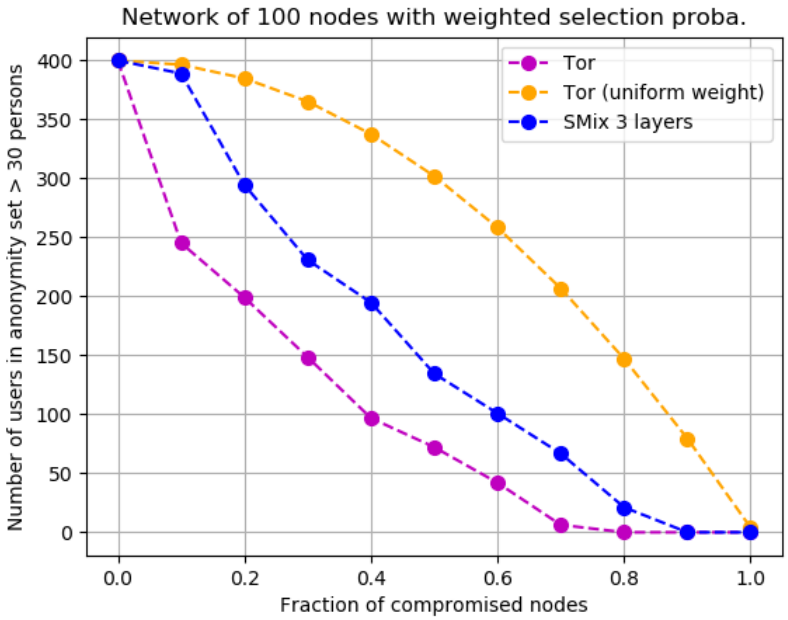


Figure 5.2: Anonymity comparison between Tor, both with weighted and uniform selection probabilities, and a synchronous mix network. The comparison is taken for the same total time before sending and receiving the response and the attack model is the one of section 4.3.

5.1.3 Comparing multiple search engines

Figure 5.3 represents the required latency and number of users to get an end-to-end anonymity set size of 2000 users, in comparison with some privacy preserving search engines. We observe that while DuckDuckGo has a sufficient query rate to bring this anonymity set size, the other are not popular enough. Also note that the mixing latency becomes constant at around 300 users, this is due to the fact that after this point, the mix have attain their full capacity and thus other mixes need to be added but it does not increase the latency for one request. This full capacity of 300 users was based on a comparison with Tor, which as of May 2019, has approximately 2M users and 6000 relays, giving approximatively 330 users per relay.

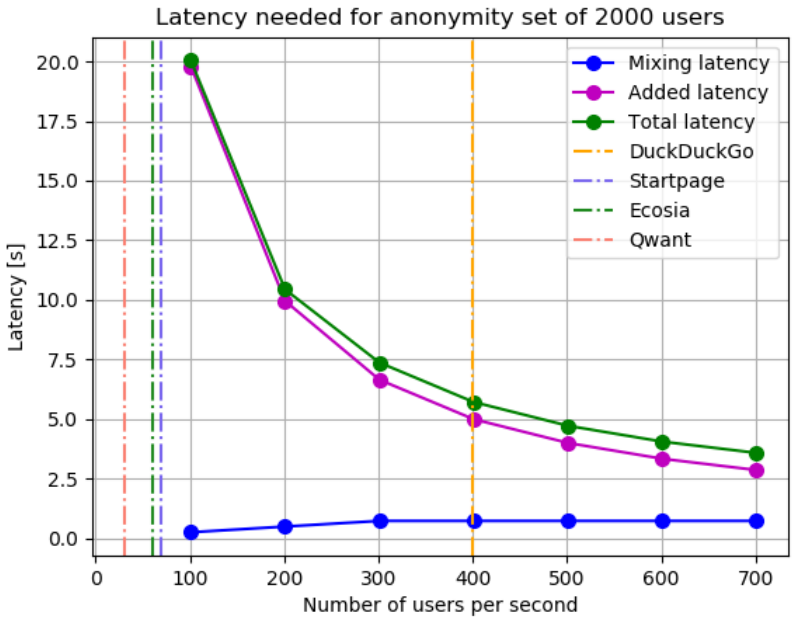


Figure 5.3: Latency added due to the mixing and delaying as functions of number of users per second on the network for an anonymity set of size 1500.

Let us now evaluate the bandwidth that would be required for implementing a mix network for DuckDuckGo. With a query rate of 400 requests per second, with a desired anonymity set of 2000 users, we need at least 5 seconds of total latency.

For the *sending* part we have a highest rate of $5 * 400$ requests sent to the first layer of mixes per second, and of that $1 * 400$ will be real and the rest dummy. With sending requests of size $B_q = 1\text{kB}$, we have

$$5 * 400/s * 1\text{kB} \leq b_i * w \qquad 500\text{kB}\,\text{s}^{-1} \leq b_i$$

For the *receiving* part, the same applies with responses of $B_r = 50\text{kB}$

$$5 * 400\text{s}^{-1} * 50\text{kB} \leq b_i * w \qquad 25\text{MB}\,\text{s}^{-1} = 100\text{Mbit}\,\text{s}^{-1} \leq b_i$$

which is suitable for the 25% largest Tor relays.

In this first section we evaluated what a weighted selection path strategy would imply on the anonymity of the system, based on the distribution of Tor relays. The result on the latency and anonymity required makes us confident that such mix networks have a great potential in providing better anonymity while maintaining acceptable latency values.

5.2 Building a tool for scaling mix networks

In order to compare the different websites capabilities, we wanted to have a tool for getting the optimal scaling and design of mix networks when inputs such as the number of users, available servers and bandwidth were given. We therefore implemented a version which[1] behaves currently in the following way.

```
$ python simulate_mix.py -u 400 -n 4 -l 2 -tl 1 -ts 1 -s 3
> (F) number paths =  12
> (C) number paths =  2
> (s) number paths =  4
> anon. set when user-server obs. =  2000
> (F) anon. set when (1-1) compromised =  33
> (C) anon. set when (1-1) compromised =  200
> (S) anon. set when (1-1) compromised =  100
>
> req. bandwidth per relay [sending] =  68.0 MB/s
> req. bandwidth per relay [receiving] =  340.0 MB/s
```

5.3 Panoramix project

The Panoramix project[2] is a project coordinated between nine universities in Europe and funded by the European Commission[3]. The project's full name is *Privacy and Accountability in Networks via Optimized Randomized Mix-nets* and it was active from September 2015 to January 2019. Its main objective was to develop an infrastructure of mix networks to enhance privacy-preserving communications and integrate it into applications that can be used by European businesses. Since a considerable time of this thesis work has been dedicated to extending and improving Panoramix implemented software, it appears important to describe the four main goals of the project.

1. Create an *open-source codebase* and infrastructure for mix networks in Europe.
2. Provide robust and verifiable *private electronic voting* protocols using mix networks (scale of 100k – −1M ballots).
3. Support *private data gathering* and cloud data handling to compile private surveying and statistics (e.g. smart city big data) with around 1M daily updates.
4. Endorse *private messaging* for multi-users communications scaling up from 90k to 200k users.

[1] https://github.com/jdewasseige/master-thesis_scale-mix-networks-tool
[2] [https://panoramix-project.eu/]
[3] [https://cordis.europa.eu/project/rcn/194872/factsheet/fr]

Objectives **2.** and **4.** have led to the implementation of the Panoramix API [Pan] which provides functionality to create and operate mix networks. It is written in PYTHON and its main focus is the implementation electronic voting protocols. Our goal was thus to extend it to web communication and add latency in the system, as the current system only relies on the cryptographic operations and do not consider security against traffic analysis attacks. We first review the implementation infrastructure, then describe our contributions and finally present the determinant results and explain how they provide insights into the designing and scaling of mix networks for web practices.

5.3.1 Overview of implementation

The two main components of the implementation are *peers* exchanging *messages*.

Any participant in the mix network can be a peer. Types of participants in the mix network include contributors, correspondent and auditors. Each peer is registered in the mix network via a cryptographic identifier.

The messages are deposed to *endpoints* which process the messages by batch. Each endpoint has an *inbox* to which the messages are sent and when the number of messages goes beyond some chosen threshold, the peer first collects the messages, then it does the processing and mixing, and finally posts them to its *outbox*.

The system runs with a Flask app, contained in `agent.py`, allowing to visualize the endpoints, messages and cycles on a local server. The end points are represented with a RestClient object. The messages are posted and retrieved from the endpoints using the `requests` package which allow to make HTTP requests easily.

A core part of the system is the `SphinxmixClient` which contains the peers, endpoints, cycles and messages (each of them is a RestClient). This object contains the method to handle the reception of messages. The user then interacts with the `panoramix_agent`, which itself uses the SphinxmixClient.

For example, in order to setup a sphinxmix mix network, the following can be done:

- Set up the server, which is an SQLite database, and start it;
- create the sphinxmix mixers, by running the `panoramix-wizard`, each in a different terminal (the list of peers has to be specified);
- the mix network then for the messages to be delivered at the address

  `http://127.0.0.1:8000/panoramix/endpoints/<mixnet_name>/`

  and will later on output the mixed messages at the following endpoint;

  `http://127.0.0.1:8000/panoramix/endpoints/<mixnet_name>_output/`
- the end user can be created in another terminal by running `sphimxmix-agent`.

5.3.2 Contributions

The contributions to the main repository are twofold. First, it includes work that was sent by mail to the maintainer of the repository which he then pushed himself (mainly errors to make the system run). Second, modifications to the API's official repository [4] include updating the messages packet format with the `sphinx` specification [DG09]. Another error in the way the group generator g was exponentiated was fixed.

Then, with the goal of extending the use of the API to synchronous mix networks with search traffic websites we worked on a copy of repository[5]. We managed to make the system add latency but the system is still not practically usable for directly making web requests.

5.3.3 Results

Beside contributing to the repository, an interesting result for designing practically mix networks was obtained. We observed the time taken by a number of messages to be sent from the mix network and compared it with the time taken to decrypt, mix these same messages. The results obtained are contained in Figure 5.4, giving an optimal number of messages of around 100 to 150 messages when no precomputation is done.

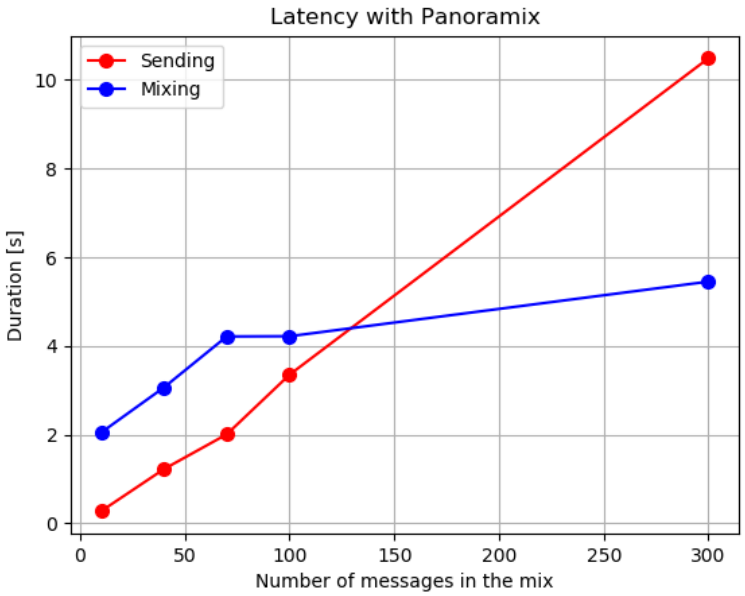


Figure 5.4: Sending and mixing overhead latency of one mix in Panoramix.

[4] `https://github.com/grnet/panoramix`
[5] `https://github.com/jdewasseige/panoramix`

Conclusion

The increasing amount of data collected about users on the Internet, and revelations that such data has been used for commercial, juridic and political purposes, without the users consent, have raised more than ever the need for networks allowing to access information in a privacy preserving way. In an attempt to provide better anonymity to users than what they are currently able to get from state of the art systems, this thesis considered the problem of using synchronous mix networks for high traffic web communication with small requests sizes. By focusing on such traffic, we hoped to leverage the resistance of mix networks to end-to-end correlation attacks, while maintaining acceptable latency for interactive use.

In Chapter 4, we considered how multiple parameters could influence the design of mix networks. The first one was the topology of the network, which we characterized using two metrics: number of paths and entropy. Our results showed that while free routes networks had a higher number of paths, their entropy was actually lower than stratified networks, for a given number of mixes. This was explained by a larger number of messages being sent to the mixes in each layer in stratified networks than for free routes. However, these measures did not take into account the average number of messages on each path, which also needs to be high to avoid intersection attacks. This allowed us to formulate two conditions for the dummy traffic needed: largest path set and timing indistinguishability. Based on these conditions, we were able to analyze the usefulness of dummy traffic and its relation with the anonymity set of the users. In the last part of this chapter, we built a realistic adversarial model based on an optimal choice of nodes to compromise from the adversary's perspective. Putting together all the studied parameters, we then evaluated mix networks for different parameters values against the adversarial model using an anonymity metric proposed in Chapter 1. The results obtained show that synchronous mix networks are a viable solution to improve anonymity as long as the assumptions for the specific type of traffic considered hold true.

In Chapter 5 we analyzed to what extent the resources needed to build the mix networks were realistic. By using data of Tor's current relays bandwidth capabilities, we were able to get a realistic distribution of expected bandwidth capabilities. Based on this, we compared how it changed the results obtained in the previous chapter. The results showed that by compromising 10% of the largest relays, our attack succeeded in compromising 37.5%

of the users, while it only compromised around 1% of users in the (fictional) case of uniform distributed relays. We then compared DuckDuckGo to other supposedly privacy-preserving search engines: Qwant, Startpage and Ecosia, and concluded that their current query rate was too small to provide enough anonymity while keeping an acceptable latency overhead of less than 10 seconds. Our contributions to the Panoramix API allowed us to compare the time required to shuffle and decrypt messages, to the time needed to send them. It was shown that the optimal capacity was between 100 and 150 messages.

As a meeting point of all the obtained results, the use of synchronous mix networks to the specific type of traffic considered is promising. Indeed, using relays with bandwidth capacities superior to $100\text{Mbit}\,\text{s}^{-1}$ lead to a viable solution for DuckDuckGo, bringing better resistance against traffic confirmation than Tor, while keeping latency overhead values between 6 and 10 seconds. However, we have to keep in mind that if such an anonymity network is deployed for a specific type of traffic, a global attacker will be able to know that the user is getting content from those websites. This question of *membership* is to be considered by the users themselves: Is the cost of disclosing the kind of web sites one visits worth getting stronger resistance against traffic confirmation? Depending on the user, both positive and negative answers to this question are acceptable and therefore leaves a large potential for the future of synchronous mix networks for web communication.

As further work, we propose two directions. First, extending the Panoramix API so that it handles real logs of user traffic would be a step towards effectively modeling realistic traffic confirmation. This would also improve the precision and realism of the security analysis of mix networks we conducted. Second, in an attempt to have more control over the number of data packets in the network and to reduce the dummy traffic overhead, an interesting idea could be to consider splitting the response packets into multiple smaller packets, sending them through independent paths, and allowing a non ordered reception on the user side. Appendix A contains a more detailed explanation of how this idea could be put into application and includes potential advantages.

Finally, for the last lines of this thesis, I want to bring forward an interesting and disturbingly funny point raised by Dingledine, one of the founders of Tor. "*Weak security allow to have low latency; low latency means more users; and more users leads to stronger security.*" While this is obviously something which does not have to be taken seriously, it points out that for anonymity networks to be effective, having a large amount of users is very important, making the adoption criterion such as very low latency a crucial design point. Research on what precise amount of latency should be decreased to balance an increase in the number of users surely is a difficult question to answer but opens up interesting research directions.

Appendix

Appendix A

Splitting response packets

This appendix explains an idea for further work considerations. The idea applies consists in using the response request structure in the following way: the server split the response into multiple independent packets. Let us apply this idea with search engines. Each response contains a fixed number of titles and URLs, and by sending them in multiple independent packets through the mix network, we allow them to come at different time intervals to the user, and thus take different paths through the mixes. The sending of a query by the user proceeds as usual in a mix environment, together with dummy packets send to balance the number of packets received from the server (see A.1 for a schematic representation of the sending part).

Current mix solutions either keep responses in one packet, or break them up into smaller packets but require them to be reassembled in the last mix before the user gets it (like in Mixmaster remailer) because the user need all the data contained in one packet for it to be "usable". In the case of web search results, one could allow for the titles and URLs to arrive at different moments (in tenth of seconds intervals) (see A.2 for an example of how a search response could be split). This could lead to the following improvements

1. increase number of messages in the mix network while keeping the same amount of "useful" data
2. reduce end-to-end traffic correlation since one need to monitor multiple independent packets with overhead latency in-between them
3. in case of system failure, the user would still get a usable response, and the system could use non delivered packets to identify corrupted mixes/paths

Potential downsides would be that the user might sometimes not receive the proposed URLs in the expected order (see A.3 for a schematic representation of the response part).

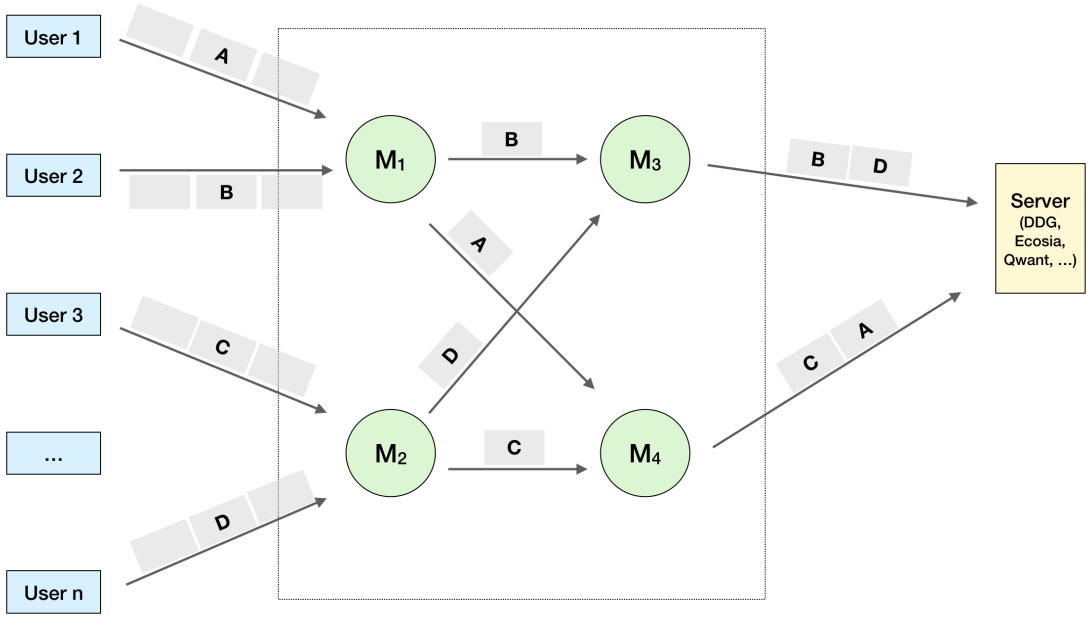


Figure A.1: Schema of four users sending queries to server through mix network to illustrate the idea of splitting.

Figure A.2: Example of how a search request response could be split into smaller packets, each containing results.

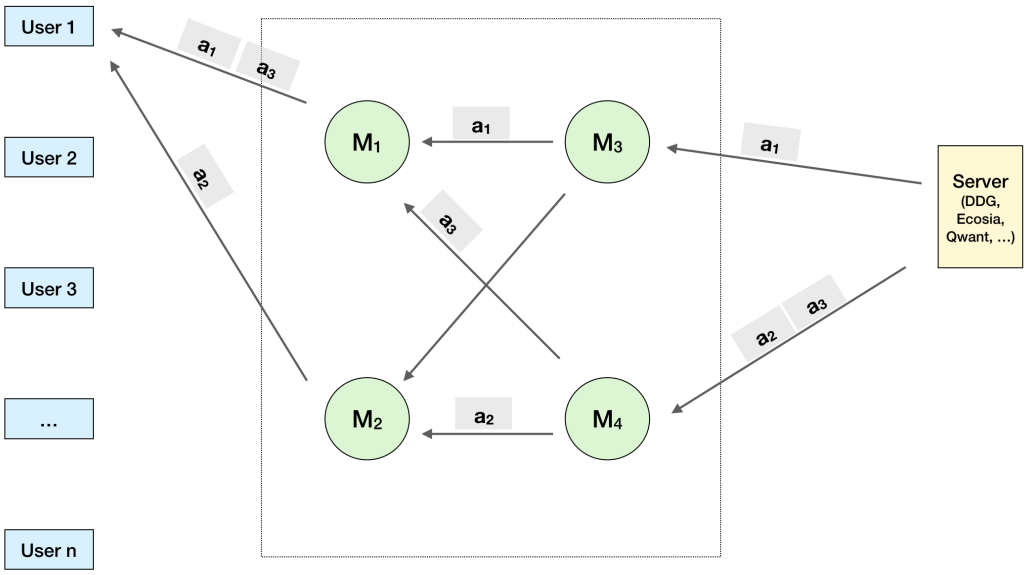


Figure A.3: Schema of server sending split response to users through mix network to illustrate the idea of splitting. The multiple packets are independent and might arrive in a non ordered.

List of Figures

List of Tables

Bibliography

[Abb+07] Timothy G Abbott *et al.* "Browser-based attacks on Tor". In: *International Workshop on Privacy Enhancing Technologies*. Springer. 2007, pp. 184–199. DOI: 10.1007/978-3-540-75551-7_12.

[AG16] Mashael Alsabah and Ian Goldberg. "Performance and Security Improvements for Tor". In: *ACM Computing Surveys* 49.2 (Sept. 2016), pp. 1–36. DOI: 10.1145/2946802.

[BAG12] Michael Brennan, Sadia Afroz, and Rachel Greenstadt. "Adversarial stylometry". In: *ACM Transactions on Information and System Security* 15.3 (Nov. 2012), pp. 1–22. DOI: 10.1145/2382448.2382450.

[Bau+14] Zygmunt Bauman *et al.* "After Snowden: Rethinking the Impact of Surveillance". In: *International Political Sociology* 8.2 (May 2014), pp. 121–144. DOI: 10.1111/ips.12048.

[Bau11] Kevin Scott Bauer. "Improving Security and Performance in Low Latency Anonymous Networks". PhD thesis. University of Denver, 2011.

[BFK00] Oliver Berthold, Hannes Federrath, and Marit Köhntopp. "Project "anonymity and unobservability in the Internet"". In: *Proceedings of the tenth conference on Computers, freedom and privacy challenging the assumptions - CFP '00*. ACM Press, 2000. DOI: 10.1145/332186.332211.

[BFK01] Oliver Berthold, Hannes Federrath, and Stefan Köpsell. "Web MIXes: A System for Anonymous and Unobservable Internet Access". In: *Designing Privacy Enhancing Technologies*. Springer Berlin Heidelberg, 2001, pp. 115–129. DOI: 10.1007/3-540-44702-4_7.

[BL03] Oliver Berthold and Heinrich Langos. "Dummy Traffic against Long Term Intersection Attacks". In: *Privacy Enhancing Technologies*. Springer Berlin Heidelberg, 2003, pp. 110–128. DOI: 10.1007/3-540-36467-6_9.

[BMS01] Adam Back, Ulf Möller, and Anton Stiglic. "Traffic Analysis Attacks and Trade-Offs in Anonymity Providing Systems". In: *Information Hiding*. Springer Berlin Heidelberg, 2001, pp. 245–257. DOI: 10.1007/3-540-45496-9_18.

[BPS01] Oliver Berthold, Andreas Pfitzmann, and Ronny Standtke. “The Disadvantages of Free MIX Routes and How to Overcome Them”. In: *Designing Privacy Enhancing Technologies*. Springer Berlin Heidelberg, 2001, pp. 30–45. DOI: 10.1007/3-540-44702-4_3.

[Cai+12] Xiang Cai *et al.* “Touching from a distance”. In: *Proceedings of the 2012 ACM conference on Computer and communications security - CCS '12*. ACM Press, 2012. DOI: 10.1145/2382196.2382260.

[Cam99] Duncan Campbell. “Development of surveillance technology and risk of abuse of economic information. The state of the art in communications Intelligence (COMINT) of automated processing for intelligence purposes of intercepted broadband multi-language leased or common carrier systems, and its applicability to COMINT targeting and selection, including speech recognition. PE 168.184/Vol 2/5”. In: (1999).

[CD07] Richard Clayton and George Danezis. “Introducing Traffic Analysis”. In: *Digital Privacy*. Auerbach Publications, Dec. 2007, pp. 95–116. DOI: 10.1201/9781420052183.ch5.

[Cha+14] Sambuddho Chakravarty *et al.* “On the Effectiveness of Traffic Analysis against Anonymity Networks Using Flow Records”. In: *Passive and Active Measurement*. Springer International Publishing, 2014, pp. 247–257. DOI: 10.1007/978-3-319-04918-2_24.

[Cha+17] David Chaum *et al.* “cMix: Mixing with Minimal Real-Time Asymmetric Cryptographic Operations”. In: *Applied Cryptography and Network Security*. Springer International Publishing, 2017, pp. 557–578. DOI: 10.1007/978-3-319-61204-1_28.

[Cha81] David L. Chaum. “Untraceable electronic mail, return addresses, and digital pseudonyms”. In: *Communications of the ACM* 24.2 (Feb. 1981), pp. 84–90. DOI: 10.1145/358549.358563.

[Cha88] David Chaum. “The dining cryptographers problem: Unconditional sender and recipient untraceability”. In: *Journal of Cryptology* 1.1 (1988). DOI: 10.1007/bf00206326.

[Con19] Josh Constine. *Facebook pays teens to install VPN that spies on them*. Feb. 2019. URL: https://techcrunch.com/2019/01/29/facebook-project-atlas/.

[CVH09] Jordi Castellà-Roca, Alexandre Viejo, and Jordi Herrera-Joancomartí. “Preserving user's privacy in web search engines”. In: *Computer Communications* 32.13-14 (Aug. 2009), pp. 1541–1551. DOI: 10.1016/j.comcom.2009.05.009.

[Dan+10] George Danezis *et al.* "Drac : An Architecture for Anonymous Low-Volume Communications". In: *Privacy Enhancing Technologies*. Springer Berlin Heidelberg, 2010, pp. 202–219. DOI: 10.1007/978-3-642-14527-8_12.

[Dan03] George Danezis. "Mix-networks with restricted routes". In: *International Workshop on Privacy Enhancing Technologies*. Springer. 2003, pp. 1–17. DOI: 10.1007/978-3-540-40956-4_1.

[Dan04] George Danezis. "Better Anonymous Communications". PhD thesis. University of Cambridge, 2004.

[Dan05] George Danezis. "The Traffic Analysis of Continuous-Time Mixes". In: *Privacy Enhancing Technologies*. Springer Berlin Heidelberg, 2005, pp. 35–50. DOI: 10.1007/11423409_3.

[Das+18] Debajyoti Das *et al.* "Anonymity Trilemma: Strong Anonymity, Low Bandwidth Overhead, Low Latency-Choose Two". In: *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE. May 2018, pp. 108–126. DOI: 10.1109/sp.2018.00011.

[DDM] G. Danezis, R. Dingledine, and N. Mathewson. "Mixminion: design of a type III anonymous remailer protocol". In: *Proceedings 19th International Conference on Data Engineering (Cat. No.03CH37405)*. IEEE Comput. Soc. DOI: 10.1109/secpri.2003.1199323.

[DFM01] Roger Dingledine, Michael J Freedman, and David Molnar. "The free haven project: Distributed anonymous storage service". In: *Designing Privacy Enhancing Technologies*. Springer. 2001, pp. 67–95. DOI: 10.1007/3-540-44702-4_5.

[DG09] George Danezis and Ian Goldberg. "Sphinx: A Compact and Provably Secure Mix Format". In: *2009 30th IEEE Symposium on Security and Privacy*. IEEE, May 2009. DOI: 10.1109/sp.2009.15.

[Dia05] Claudia Diaz. "Anonymity and Privacy in Electronic Services". PhD thesis. Katholieke Universiteit Leuven, 2005.

[Din00] Roger Dingledine. "The free haven project: Design and deployment of an anonymous secure data haven". PhD thesis. Massachusetts Institute of Technology, 2000.

[DM09] Roger Dingledine and Steven J Murdoch. "Performance Improvements on Tor or, Why Tor is slow and what we're going to do about it". In: (2009). URL: http://www.torproject.org/press/presskit/2009-03-11-performance.pdf.

[DMS04] Roger Dingledine, Nick Mathewson, and Paul Syverson. *Tor: The second-generation onion router*. Tech. rep. Naval Research Lab Washington DC, Jan. 2004. DOI: 10.21236/ada465464.

[DMT10] Claudia Diaz, Steven J Murdoch, and Carmela Troncoso. "Impact of network topology on anonymity and overhead in low-latency anonymity networks". In: *International Symposium on Privacy Enhancing Technologies Symposium*. Springer. 2010, pp. 184–201. DOI: 10.1007/978-3-642-14527-8_11.

[DP04] Claudia Diaz and Bart Preneel. "Taxonomy of Mixes and Dummy Traffic". In: *Information Security Management, Education and Privacy*. Kluwer Academic Publishers, 2004, pp. 217–232. DOI: 10.1007/1-4020-8145-6_18.

[DS03] Claudia Diaz and Andrei Serjantov. "Generalising Mixes". In: *Privacy Enhancing Technologies*. Springer Berlin Heidelberg, 2003, pp. 18–31. DOI: 10.1007/978-3-540-40956-4_2.

[DSS05] Roger Dingledine, Vitaly Shmatikov, and Paul Syverson. "Synchronous batching: From cascades to free routes". In: *International Workshop on Privacy Enhancing Technologies*. Springer. 2005, pp. 186–206. DOI: 10.1007/11423409_12.

[Eck10] Peter Eckersley. "How unique is your web browser?" In: *International Symposium on Privacy Enhancing Technologies Symposium*. Springer. 2010, pp. 1–18. DOI: 10.1007/978-3-642-14527-8_1.

[EDG09] Nathan S Evans, Roger Dingledine, and Christian Grothoff. "A Practical Congestion Attack on Tor Using Long Paths". In: *USENIX Security Symposium*. 2009, pp. 33–50.

[EFF] Electronic Frontier Foundation (EFF). *Mandatory Data Retention*. URL: https://www.eff.org/issues/mandatory-data-retention.

[FM02] Michael J. Freedman and Robert Morris. "Tarzan: A Peer-to-peer Anonymizing Network Layer". In: *Proceedings of the 9th ACM Conference on Computer and Communications Security*. CCS '02. Washington, DC, USA: ACM, 2002, pp. 193–206. ISBN: 1-58113-612-9. DOI: 10.1145/586110.586137.

[FR14] *Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing*. Tech. rep. June 2014. DOI: 10.17487/rfc7230. URL: https://doi.org/10.17487%2Frfc7230.

[GM13] Glenn Greenwald and Ewen MacAskill. "NSA Prism program taps into user data of Apple, Google and others". In: *The Guardian* 7.6 (2013), pp. 1–43.

[Gol+04] Philippe Golle *et al.* "Universal Re-encryption for Mixnets". In: *Topics in Cryptology – CT-RSA 2004*. Springer Berlin Heidelberg, 2004, pp. 163–178. DOI: 10.1007/978-3-540-24660-2_14.

[GRS96] David M. Goldschlag, Michael G. Reed, and Paul F. Syverson. “Hiding Routing information”. In: *Information Hiding*. Springer Berlin Heidelberg, 1996, pp. 137–150. DOI: 10.1007/3-540-61996-8_37.

[GT96] Ceki Gulcu and Gene Tsudik. “Mixing E-mail with Babel”. In: *Network and Distributed System Security, 1996., Proceedings of the Symposium on*. IEEE. 1996, pp. 2–16. DOI: 10.1109/ndss.1996.492350.

[Hel96] Johan Helsingius. “Johan Helsingius gets injunction in scientology case privacy protection of anonymous messages still unclear”. In: (Sept. 1996). URL: https://web.archive.org/web/20050921022510/http://www.eff.org/Privacy/Anonymity/960923_penet_injunction.announce.

[HL10] Carmit Hazay and Yehuda Lindell. *Efficient secure two-party protocols: Techniques and constructions*. Springer Science & Business Media, 2010. DOI: 10.1007/978-3-642-14303-8.

[HVC10] Nicholas Hopper, Eugene Y. Vasserman, and Eric Chan-TIN. “How much anonymity does network latency leak?” In: *ACM Transactions on Information and System Security* 13.2 (Feb. 2010), pp. 1–28. DOI: 10.1145/1698750.1698753.

[JJ01] Markus Jakobsson and Ari Juels. “An optimally robust hybrid mix network”. In: *Proceedings of the twentieth annual ACM symposium on Principles of distributed computing*. ACM. 2001, pp. 284–292. DOI: 10.1145/383962.384046.

[Joh+13] Aaron Johnson *et al.* “Users get routed: Traffic correlation on Tor by realistic adversaries”. In: *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. ACM. 2013, pp. 337–348. DOI: 10.1145/2508859.2516651.

[KEB98] Dogan Kesdogan, Jan Egner, and Roland Büschkes. “Stop- and- Go-MIXes Providing Probabilistic Anonymity in an Open System”. In: *Information Hiding*. Springer Berlin Heidelberg, 1998, pp. 83–98. DOI: 10.1007/3-540-49380-8_7.

[KL08] Foaad Khosmood and Robert A Levinson. “Automatic natural language style classification and transformation”. In: *BCS Corpus Profiling Workshop, London, UK*. sn. 2008.

[Lev+04] Brian N. Levine *et al.* “Timing Attacks in Low-Latency Mix-Based Systems”. In: *Proceedings of Financial Cryptography (FC ’04)*. Ed. by Ari Juels. Springer-Verlag, LNCS 3110, Feb. 2004, pp. 251–265. DOI: 10.1007/978-3-540-27809-2_25.

[Lin+11] Zhen Ling *et al.* “Equal-sized cells mean equal-sized packets in Tor?” In: *Communications (ICC), 2011 IEEE International Conference on*. IEEE. June 2011, pp. 1–6. DOI: 10.1109/icc.2011.5962653.

[LMD10] Karsten Loesing, Steven J. Murdoch, and Roger Dingledine. “A Case Study on Measuring Statistical Data in the Tor Anonymity Network”. In: *Financial Cryptography and Data Security*. Springer Berlin Heidelberg, 2010, pp. 203–215. DOI: 10.1007/978-3-642-14992-4_19.

[Lom18] Natasha Lomas. *Pro-privacy search engine DuckDuckGo hits 30M daily searches, up 50% in a year*. Oct. 2018. URL: https://techcrunch.com/2018/10/11/pro-privacy-search-engine-duckduckgo-hits-30m-daily-searches-up-50-in-a-year/.

[Mas13] Mike Masnick. *ISP CEO Explains What Happens When The NSA Shows Up At Your Door*. July 2013. URL: https://www.techdirt.com/articles/20130722/00303923879/isp-ceo-explains-what-happens-when-nsa-shows-up-your-door.shtml.

[MC10] Aleecia M McDonald and Lorrie Faith Cranor. “Americans’ attitudes about internet behavioral advertising practices”. In: *Proceedings of the 9th annual ACM workshop on Privacy in the electronic society*. ACM. 2010, pp. 63–72.

[MD04] Nick Mathewson and Roger Dingledine. “Practical traffic analysis: Extending and resisting statistical disclosure”. In: *International Workshop on Privacy Enhancing Technologies*. Springer. 2004, pp. 17–34. DOI: 10.1007/11423409_2.

[MD05] Steven J Murdoch and George Danezis. “Low-cost traffic analysis of Tor”. In: *Security and Privacy, 2005 IEEE Symposium on*. IEEE. 2005, pp. 183–195. DOI: 10.1109/sp.2005.12.

[Moc01] Willson Mock. *How the Internet Works (A Simple Explanation)*. Oct. 201. URL: https://medium.com/@fay_jai/how-the-internet-works-a-simple-explanation-ca8053c71661.

[Möl03] Bodo Möller. “Provably Secure Public-Key Encryption for Length-Preserving Chaumian Mixes”. In: *Topics in Cryptology — CT-RSA 2003*. Springer Berlin Heidelberg, 2003, pp. 244–262. DOI: 10.1007/3-540-36563-x_17.

[MS02] David Martin and Andrew Schulman. *Deanonymizing users of the safeweb anonymizing service*. Tech. rep. Boston University Computer Science Department, 2002.

[Mül+12] Sebastian Müller *et al.* “Distributed Performance Measurement and Usability Assessment of the Tor Anonymization Network”. In: *Future Internet* 4.2 (May 2012), pp. 488–513. DOI: 10.3390/fi4020488.

[Mül15] Konstantin Müller. “Defending end-to-end confirmation attacks against the tor network”. MA thesis. Gjøvik University College, 2015.

[MW08] Steven J Murdoch and Robert NM Watson. “Metrics for security and performance in low-latency anonymity systems”. In: *International Symposium on Privacy Enhancing Technologies Symposium*. Springer. 2008, pp. 115–132. DOI: 10.1007/978-3-540-70630-4_8.

[MZ07] Steven J. Murdoch and Piotr Zieliński. “Sampled Traffic Analysis by Internet-Exchange-Level Adversaries”. In: *Proceedings of the Seventh Workshop on Privacy Enhancing Technologies (PET 2007)*. Ed. by Nikita Borisov and Philippe Golle. Ottawa, Canada: Springer, June 2007. DOI: 10.1007/978-3-540-75551-7_11.

[OB09] Gavin O’Gorman and Stephen Blott. “Improving stream correlation attacks on anonymous networks”. In: *Proceedings of the 2009 ACM symposium on Applied Computing*. ACM. 2009, pp. 2024–2028. DOI: 10.1145/1529282.1529732.

[Ola18] Joseph Ola. *Privacy Violations by free VPN service providers*. June 2018. URL: https://iapp.org/news/a/privacy-violations-by-free-vpn-service-providers/.

[Pan] Panoramix project. *Panoramix*. https://github.com/grnet/panoramix.

[PH08] Andreas Pfitzmann and Marit Hansen. “Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management-a consolidated proposal for terminology”. In: *Version v0* 31 (2008), p. 15.

[PIK93] Choonsik Park, Kazutomo Itoh, and Kaoru Kurosawa. “Efficient anonymous channel and all/nothing election scheme”. In: *Workshop on the Theory and Application of of Cryptographic Techniques*. Springer. 1993, pp. 248–259. DOI: 10.1007/3-540-48285-7_21.

[PS11] Sai Teja Peddinti and Nitesh Saxena. “On the effectiveness of anonymizing networks for web search privacy”. In: *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*. ACM. 2011, pp. 483–489. DOI: 10.1145/1966913.1966984.

[PW87] Andreas Pfitzmann and Michael Waidner. “Networks without user observability”. In: *Computers & Security* 6.2 (Apr. 1987), pp. 158–166. DOI: 10.1016/0167-4048(87)90087-3.

[Ray01] Jean-François Raymond. “Traffic Analysis: Protocols, Attacks, Design Issues, and Open Problems”. In: *Designing Privacy Enhancing Technologies*. Springer Berlin Heidelberg, 2001, pp. 10–29. DOI: 10.1007/3-540-44702-4_2.

[Roe10] Michael Roe. *Cryptography and evidence*. Tech. rep. University of Cambridge, Computer Laboratory, 2010.

[RP02] Marc Rennhard and Bernhard Plattner. “Introducing MorphMix: peer-to-peer based anonymous Internet usage with collusion detection”. In: *Proceedings of*

the 2002 ACM workshop on Privacy in the Electronic Society. ACM. 2002, pp. 91–102. DOI: 10.1145/644527.644537.

[RP03] Marc Rennhard and Bernhard Plattner. "Practical anonymity for the masses with mix-networks". In: *WET ICE 2003. Proceedings. Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2003.* IEEE. 2003, pp. 255–260. DOI: 10.1109/enabl.2003.1231417.

[RP17] Florentin Rochet and Olivier Pereira. "Waterfilling: Balancing the Tor network with maximum diversity". In: *Proceedings on Privacy Enhancing Technologies* 2017.2 (Apr. 2017), pp. 4–22. DOI: 10.1515/popets-2017-0013.

[RR98] Michael K. Reiter and Aviel D. Rubin. "Crowds: Anonymity for Web Transactions". In: *ACM Trans. Inf. Syst. Secur.* 1.1 (Nov. 1998), pp. 66–92. ISSN: 1094-9224. DOI: 10.1145/290163.290168.

[RSG98] Michael G Reed, Paul F Syverson, and David M Goldschlag. "Anonymous connections and onion routing". In: *IEEE Journal on Selected areas in Communications* 16.4 (1998), pp. 482–494. DOI: 10.21236/ada465335.

[Sai+07] Felipe Saint-Jean *et al.* "Private web search". In: *Proceedings of the 2007 ACM workshop on Privacy in electronic society*. ACM. 2007, pp. 84–90. DOI: 10.1145/1314333.1314351.

[SD02] Andrei Serjantov and George Danezis. "Towards an information theoretic metric for anonymity". In: *International Workshop on Privacy Enhancing Technologies*. Springer. 2002, pp. 41–53. DOI: 10.1007/3-540-36467-6_4.

[SDS02] Andrei Serjantov, Roger Dingledine, and Paul Syverson. "From a Trickle to a Flood: Active Attacks on Several Mix Types". In: *Information Hiding*. Springer Berlin Heidelberg, Dec. 2002, pp. 36–52. DOI: 10.1007/3-540-36415-3_3.

[Shm02] Vitaly Shmatikov. "Probabilistic analysis of anonymity". In: *Proceedings 15th IEEE Computer Security Foundations Workshop. CSFW-15*. IEEE. 2002, pp. 119–128. DOI: 10.1109/csfw.2002.1021811.

[Sol07] Daniel J Solove. "I've got nothing to hide and other misunderstandings of privacy". In: *San Diego L. Rev.* 44 (2007), p. 745.

[SP06] K. Sampigethaya and R. Poovendran. "A Survey on Mix Networks and Their Secure Applications". In: *Proceedings of the IEEE* 94.12 (Dec. 2006), pp. 2142–2181. DOI: 10.1109/jproc.2006.889687.

[SRG00] Paul F Syverson, Michael G Reed, and David M Goldschlag. "Onion Routing access configurations". In: *Proceedings DARPA Information Survivability Conference and Exposition. DISCEX'00*. Vol. 1. IEEE. 2000, pp. 34–40. DOI: 10.1109/discex.2000.824941.

[SS03] Andrei Serjantov and Peter Sewell. “Passive Attack Analysis for Connection-Based Anonymity Systems”. In: *Proceedings of ESORICS 2003*. Oct. 2003. DOI: 10.1007/978-3-540-39650-5_7.

[STZ07] Xuehua Shen, Bin Tan, and ChengXiang Zhai. “Privacy Protection in Personalized Search”. In: *SIGIR Forum* 41.1 (June 2007), pp. 4–17. ISSN: 0163-5840. DOI: 10.1145/1273221.1273222.

[Sun+02] Qixiang Sun *et al.* “Statistical identification of encrypted web browsing traffic”. In: *Proceedings 2002 IEEE Symposium on Security and Privacy*. IEEE. 2002, pp. 19–30. DOI: 10.1109/secpri.2002.1004359.

[Swe02] Latanya Sweeney. “k-anonymity: A model for protecting privacy”. In: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 10.05 (Oct. 2002), pp. 557–570. DOI: 10.1142/s0218488502001648.

[Syv+01] Paul Syverson *et al.* “Towards an analysis of onion routing security”. In: *Designing Privacy Enhancing Technologies*. Springer. 2001, pp. 96–114.

[Syv11] Paul Syverson. “Sleeping dogs lie in a bed of onions but wake when mixed”. In: *4th Hot Topics in Privacy Enhancing Technologies (HotPETs 2011)* (2011).

[VC10] Alexandre Viejo and Jordi Castellà-Roca. “Using social networks to distort users’ profiles generated by web search engines”. In: *Computer Networks* 54.9 (June 2010), pp. 1343–1357. DOI: 10.1016/j.comnet.2009.11.003.

[WG13] Tao Wang and Ian Goldberg. “Improved website fingerprinting on tor”. In: *Proceedings of the 12th ACM workshop on Workshop on privacy in the electronic society*. ACM. 2013, pp. 201–212. DOI: 10.1145/2517840.2517851.

[Wri+02] Matthew K Wright *et al.* “An Analysis of the Degradation of Anonymous Protocols”. In: *NDSS*. Vol. 2. 2002, pp. 39–50.

[Wri+03] Matthew Wright *et al.* “Defending anonymous communications against passive logging attacks”. In: *Security and Privacy, 2003. Proceedings. 2003 Symposium on*. IEEE. 2003, pp. 28–41.

[Wri+04] Matthew K. Wright *et al.* “The predecessor attack”. In: *ACM Transactions on Information and System Security* 7.4 (Jan. 2004), pp. 489–522. DOI: 10.1145/1042031.1042032.

[Zhu+05] Ye Zhu *et al.* “On Flow Correlation Attacks and Countermeasures in Mix Networks”. In: *Privacy Enhancing Technologies*. Springer Berlin Heidelberg, 2005, pp. 207–225. DOI: 10.1007/11423409_13.

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain
Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/epl**