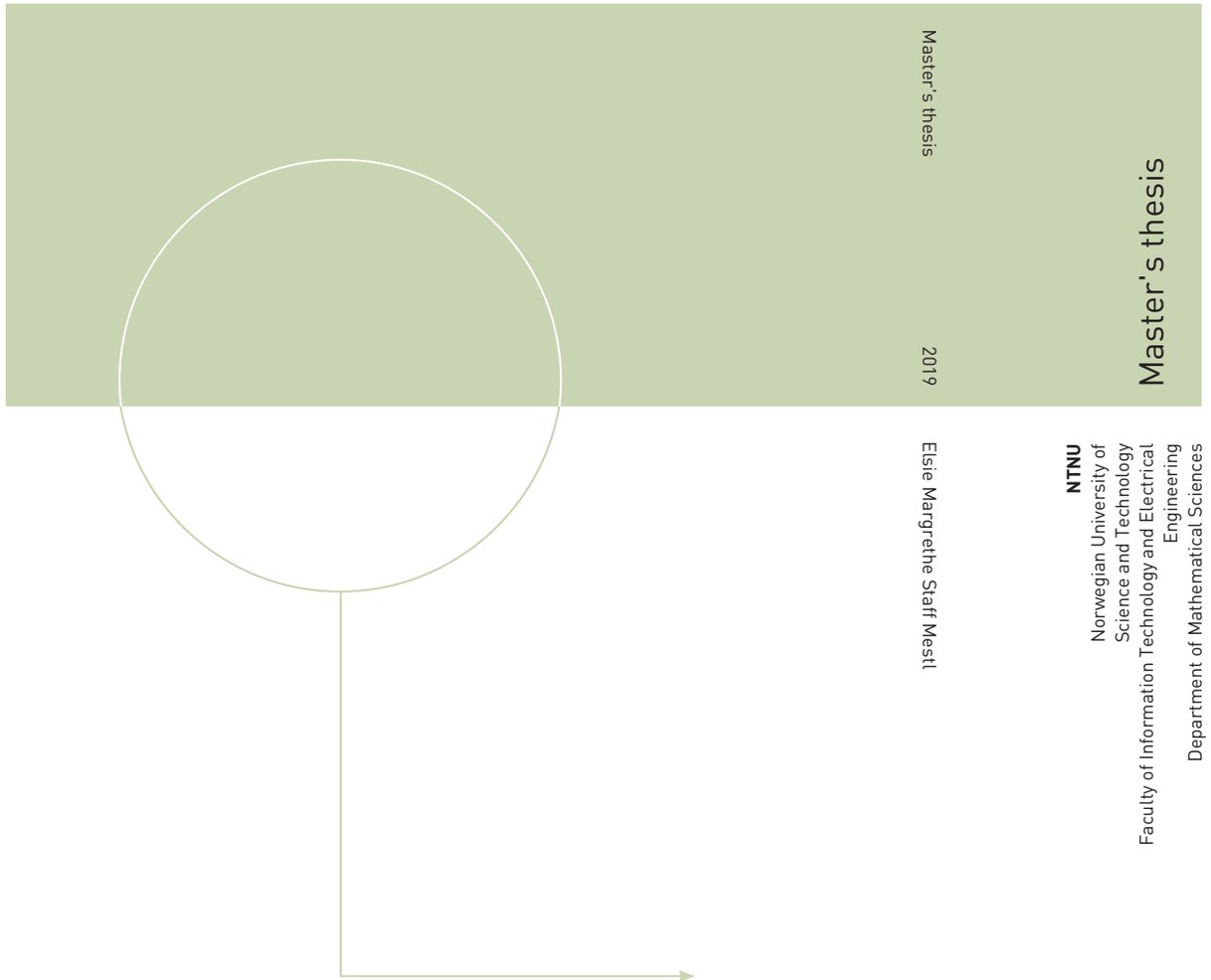


Master's thesis

2019

Elsie Margrethe Staff Mestl

NTNU
Norwegian University of
Science and Technology
Faculty of Information Technology and Electrical
Engineering
Department of Mathematical Sciences

Elsie Margrethe Staff Mestl

A Framework for Describing and Analyzing Anonymity Networks

June 2019

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of
Science and Technology

NTNU

Norwegian University of
Science and Technology

A Framework for Describing and Analyzing Anonymity Networks

Master of Mathematical Science
Submission date: June 2019
Supervisor: Kristian Gjøsteen

Norwegian University of Science and Technology
Department of Mathematical Sciences

Abstract

This thesis presents a framework allowing for an idealized model description of an anonymity network and network simulation. A main advantage of this framework is that anonymity characteristics can be defined for a specified model.

Anonymity goals and attack powers applicable to onion routing networks have been made specified, and formally defined to fit the network framework. These attack powers and anonymity goals have been related to each other in order to create a hierarchy of, respectively, power and anonymity. These anonymity goals and attack powers are then combined into anonymity notions to define an anonymity model. Via an example it is show how the anonymity framework is applied to two different anonymity networks.

To the authors know such standardized framework for analyzing anonymity networks is available in the literature.

Sammendrag

Oppgaven introduserer et rammeverk som tillater en idealisering av anonymitetesnettverk og netverksimulering. En hovedfordel med dette rammeverket er at anonyitetskarakterestikker kan bli definert for en spesifik model.

Anonymitetsmålene og angrepskreftene tilgjenelige i et løk-nettverk har blitt spesifisert, og formelt definer for å passe rammeverket. Angrepenskreftene og anonymitetsmålene har så blitt relatert til hverandre for å danne et hierarki av respektivt anonymitet og krefter. Målene og angrepene er videre kombinert for å danne anonymitetsoppfatninger (*anonymity notion*) som igjen danner en anonymitetesmodell. Gjennom et eksempel blir det vist hvordan anonymitetsrammeverket kan anvendes på to forskjellige anonymitetsnettverk.

Såvidt den undertegnede er bekjent er ingen slik modell tilgjengelig for anonymitet.

Acknowledgment

I would like to thank my supervisor, Professor Kristian Gjøsteen, who has allowed me to pursue paths that I have found interesting throughout this master writing experience, while at the same time coming with invaluable insight as to why certain paths might be difficult. I would also like to say thank you for being so easily available, and allowing me to have extra supervisions as the deadline of the thesis approached.

I would also like to extend a thanks to my parents Heidi & Thomas Mestl as well as my boyfriend Simen Keiland Fondevik for diligently proofreading the thesis and coming with great writing and structural advice.

Thank you,

Elsie Margrethe Staff Mestl

Table of Contents

1 Introduction **1**
- 1.1 Anonymity . . . 1
- 1.2 Organization of Thesis . . . 2

2 Anonymity Network Model **3**
- 2.1 Network Model . . . 3
- 2.2 Low-latency and High-latency Networks . . . 5

3 Onion Routing and Tor **7**
- 3.1 Onion Routing - the Idea . . . 7
  - 3.1.1 Replying to an Onion Message . . . 8
- 3.2 Tor . . . 10
  - 3.2.1 Circuit Construction . . . 10
  - 3.2.2 Circuit Id's . . . 11
  - 3.2.3 Multiple Connections Along one Circuit . . . 13
- 3.3 Relations between Players in an Onion Routing Network . . . 14

4 Network Simulation and Adversarial Power **15**
- 4.1 Network Simulation . . . 16
- 4.2 Passive and Active Adversaries . . . 19
- 4.3 General Low-Latency Adversarial Powers . . . 19
  - 4.3.1 Timing Attack . . . 20
  - 4.3.2 Chosen Sender Attack . . . 20
  - 4.3.3 Network Altering Attacks . . . 21
  - 4.3.4 Relating the General Attack-Powers . . . 21
- 4.4 Tor Specific Adversarial Powers . . . 22
  - 4.4.1 Timing Attack . . . 22
  - 4.4.2 Network Altering Attacks . . . 23
- 4.5 Relating Attack-Powers . . . 23

5 Anonymity Goals **25**
- 5.1 Anonymity-Specific Anonymity Goals . . . 26
  - 5.1.1 Relating Anonymity-Specific Anonymity Goals . . . 27
- 5.2 Unlinkability . . . 32
- 5.3 Anonymity and Unlinkability Relations . . . 34

6 Anonymity Notions **35**
6.1 Anonymity Games 36
6.2 Relating anonymity notions 37

7 Applications **41**
7.1 General Onion Routing 41
7.2 Mixed Network 42

8 Conclusion **45**

Bibliography **48**

Chapter 1

Introduction

The goal of an encryption scheme is in general to provide the communication parties with message integrity and confidentiality [8]. The communicating parties are, however, generally known. An anonymity network on the other hand, aims to disguise the communicating parties; message confidentiality and integrity is not something it promises.

To achieve anonymity, integrity and confidentiality, both an anonymity network as well as an encryption scheme should be employed.

1.1 Anonymity

In layman terms, anonymity is considered the ability to remain unknown, i.e. an anonymous person is someone not immediately identifiable. When someone is considered identifiable varies upon the situation, e.g. name, address or appearance. A person is considered unidentifiable when there exists no process that can find out the required identifying information about said person. The terms introduced in the next paragraph, and later formally defined in Chapter 5, are based on the definitions introduced by Pfitzmann and Hansen [17] and adapted to fit the anonymity network model introduced in Chapter 2 and Chapter 3.

In a conversation where Alice sends Bob a message, one can intuitively define Alice to have sender anonymity if she is anonymous. Similarly, Bob can be said to have receiver anonymity if he is anonymous. In addition, Alice and Bob can be considered unlinkable as long as it is not clear that they are communicating.

Obviously Alice is not anonymous with respect to herself. She might also secretly tell Bob her identity and not be anonymous with respect to him either. Still, one could say, Alice is anonymous with respect to a third observer. It is clear, therefore, that in order to speak about anonymity, it is not only important to define the anonymity type, i.e. sender or receiver, but also with respect to whom one wishes to remain anonymous.

Cryptography uses security notions to classify the security of different cryptography-schemes. These notions consist of a security goal, combined with an attack model, i.e. what we want the system is to fend against, combined with an adversarial power. First introduced by Bellare et al. [1], these notions have been further developed to contain both symmetric and asymmetric systems alike and have become a well defined standard for

analyzing new crypto-systems.

The objective of this thesis is to give a formal definition of the anonymity goals; sender anonymity, receiver anonymity and unlinkability, and formally define a framework for anonymity networks and possible adversarial powers. The thesis also looks at the anonymity notions created by these definitions, and how said notions relate.

The main work of this thesis has gone into constructing good and accurate definitions for different anonymity notions. It is desirable the give the adversary as much power as possible over the network without having the anonymity goals trivially broken. Analyzing anonymity in a system where the attacker trivially knows all the user identities is not interesting then the attacker wins a priori. However it is similarly uninteresting to analyze a system where the adversary is too constrained, as such an analysis tells nothing about how much the system can handle before anonymity is breached.

Pfitzmann and Hansen [17] introduce terminology for anonymity goals, the goals are generalized to such an extent as to work for any anonymity network model, a major part of this thesis is to specify them in such a way as to make them useful in the specific framework given in this thesis. Other papers such as "A formalization of anonymity and onion routing" by Mauw, Verschuren, and Vink [14][1], and "A Model of Onion Routing with Provable Anonymity" by Feigenbaum, Johnson, and Syverson [6][2] are more focused on analyzing onion routing in their specific models. We do not aim to analyze onion routing in the idealized framework per se, but rather to create an anonymity model that is applicable to most anonymity networks using the defined network framework.

1.2 Organization of Thesis

Chapter 2 introduces the concept of anonymity networks and a generic anonymity network model for any anonymity technique. Chapter 3 then describes a specific anonymity network method, called onion routing. Onion routing is introduced on general terms with Tor specific implementation details where needed. In Chapter 4 the adversarial powers are introduced; firstly as general anonymity network adversaries, applicable to any network model independent of message latency and anonymity technique. These adversaries are then further developed to be Tor-specific. Chapter 5 defines the anonymity goals. Both the attack powers and anonymity notions are related to each other using reductions. In Chapter 6 the anonymity notions are defined and some relations specified. Finally in Chapter 7 the framework is applied to two different networks, a general onion routing network and a simplified mix network. This both shows that the framework is applicable to network schemes other than onion routing, it also indicates the anonymity difference of different anonymity networks.

[1] The paper analyzes (sender/receiver) anonymity of a simplified onion routing system using a trace model.

[2] Feigenbaum, Johnson, and Syverson create a model for onion routing using a IO-automata, they proceed to analyze this model in order to prove when it fulfills the different anonymity requirements.

Chapter 2

Anonymity Network Model

An anonymity network always consists of a set of **players** P. These players can be subdivided into three sets. S, the set of **senders**, are the players who initiate a connection. A set R of **recipients/receivers**, they are the players the senders would like to communicate with. The final set will be denoted N, and contains players that are neither senders nor recipients but give a large enough crowd for the senders and receivers to hide among. The exact task of these remaining players vary dependent on the anonymity network and will be further expanded when needed. An example of an anonymity network can be seen in Figure 2.1 illustrating how senders send their messages into a network that somehow shuffles them around before sending the messages off to the intended recipients.

Remark. *A player in this instance is not a human, but a computer. So if person Alice would like to speak to person Bob, she instructs her computer to create a connection to Bob. Her computer then becomes the sender and Bob's computer becomes the receiver in that particular communication channel. To improve readability we will denote the sender corresponding to Alice for Alice, and similarly for receivers.*

The human Bob might not be a human at all, and might be a website bob.com instead. Again to increase readability the entity behind the receiver will be called Bob, when names are required, independent of whether it is a human or not.

In order to simplify the model, it will be assumed that S, R and N are pairwise disjoint sets. Although not entirely accurate, this choice will generally not significantly lower the anonymity, for most anonymity networks. This follows form the fact that in most cases the part a player plays in a specific communication instance is usually easy to deduce from its communication pattern. E.g. senders and receivers are simple to differentiate; senders usually sends and then (might) receive a reply, while for receivers it is the opposite. Similarly, one can differentiate senders and receivers form other players.

2.1 Network Model

An anonymity network is a network on top of an existing communication network. In other words, all players in an anonymity network are in fact clients using a communication

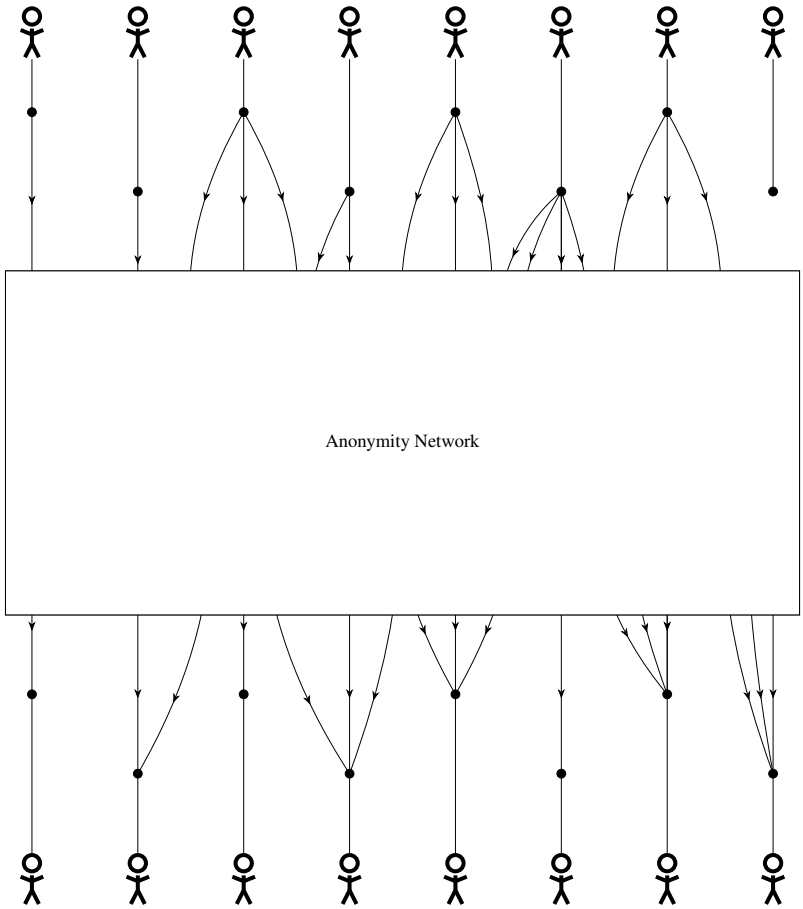


Figure 2.1: An example of an anonymity network. The nodes above the box correspond to senders, the ones below are receivers. The arrows are the paths their messages travel along. The box represents an arbitrary anonymity network and will shuffle the messages around according to its description. The aim of the *shuffling* is to confuse any observer that watches parts of the network in such a way that they will not be able to follow the message from sender to receiver.

network, i.e. players of an anonymity network are users (computers) in the underlying communication network. For readers familiar with the OSI-model, an anonymity network lives on layer 7 (application layer) of the OSI-model, whereas the network layer is layer 3 [18].

Assume a sender sends a message into the anonymity network. The message then travels between players[1] inside the anonymity network, depending on properties of that particular anonymity network, before it reaches its final destination, the receiver.

For anyone observing the communication network it appears as if the sender communicates with some player in the anonymity network while another player communicates

[1] Between mixes in a mix network, onion routers in an onion routing network, etc.

with the receiver. The intention is that by passing the message between multiple players, anyone who tries to follow the message from sender to receiver should get lost in the process.

In addition to being the set of players that are not senders nor receivers, N will also specify the network. Let, therefore, S_N, R_N and P_N denote, respectively, the sets S, R and P with respect to the network N. When there is no ambiguity as to which anonymity network is in question, the subscript will be omitted.

2.2 Low-latency and High-latency Networks

The set of all anonymity networks can be divided into two groups; **low-latency** and **high-latency** networks [12].

Low-latency networks are networks that minimize the travel time of messages through the network [12]. This is for example important if the network is to be used for time sensitive applications, such as chats. An anonymity network that uses many hours in order to send a message, and then just as long to receive a reply might not be suitable for such purposes.

The requirement for quick messages comes, however, at a cost, as the travel time of a message through the network can tell a lot about its path. For example, if whenever Alice sends a message into the network, a couple of milliseconds later a message is received by Bob, this reveals the fact that Alice and Bob are probably communicating.

High-latency networks on the other hand do not not aim to achieve minimal message delays [12]. Such networks can therefore wait until enough messages are gathered before sending them all at once. This way, the timing attack from the previous paragraph fails to reveal any information. All an attacker may grasp is that some sender is communicating with Bob. High-latency are, therefore, in general more secure than low-latency networks, but on the other hand less versatile.

In certain situations, for instance such as emails, anonymity might be valued higher than message latency. In this thesis, however, we will focus on low-latency networks when creating attack powers and anonymity goals.

Chapter 3

Onion Routing and Tor

When further developing the model, with traffic and attack notions we will have a special anonymity network in mind, namely **onion routing**. There are multiple ways an anonymity network may achieve *anonymity*, and even more platforms that offer *anonymity* using these techniques [15, 19]. This thesis will, however, focus on one specific approach named onion routing. When specific implementation details are needed they will be taken from one of the most popular onion routing networks, namely Tor [22].

It will, however, be shown in Chapter 7 that the model can be used for other anonymity networks, that are not onion based, as well. A further analysis of which anonymity networks fit the current anonymity model should be undertaken, and a further generalization may take place.

3.1 Onion Routing - the Idea

The concept of onion routing works as follows:

Say Alice wants to send Bob a message via an onion routing network N. First, she selects a minimum of three network players $or_0, \cdots, or_n \in \mathrm{N}$. In the onion routing model, network players are referred to as **onion routers**, OR. The path through the network the selected onion routers constitute, is called a **circuit**. When the sender and receiver are included it will be denoted as an **extended circuit**. The first and last OR of the circuit is often referred to as, respectively, the **entry** and **exit** router, while the remaining onion routers are **middle** routers.

Alice takes her message m and layers it in encryption. Each layer corresponds to one specific OR, and is encrypted by Alice with the key belonging to that onion router.[1] She starts with the last onion router in the circuit and ends at the first, see Figure 3.1. This multi-encrypted message is called an **onion**.

Alice then sends the onion into the network staring with or_0. The entry router decrypts the outermost encryption layer before sending the peeled onion to or_1, which peels off the next layer. This continues until all the encryption layers are peeled off and or_n sends

[1] The onion router key can be either its public key or a shared private key.

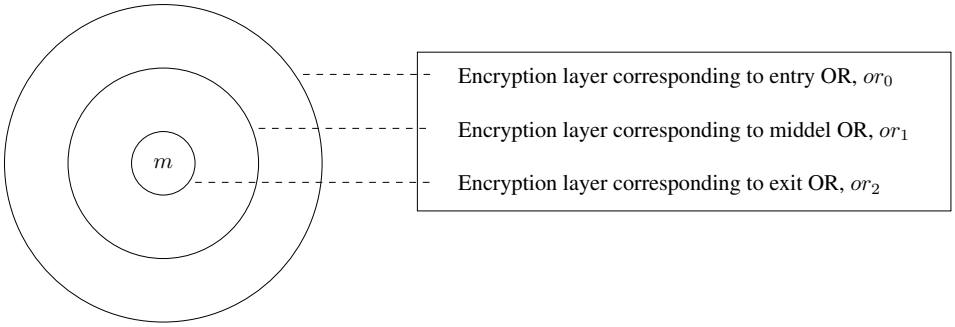


Figure 3.1: Shows an example of an onion created by the sender from message m for the circuit $or_0 - or_1 - or_2$. When the onion travels along the circuit each OR will in turn peel of the encryption layer corresponding to itself before sending it to the next player in the circuit.

the final message m to Bob.[2] Figure 3.3 shows an example of an onion being sent over a circuit.

Each onion router will in principal only know its neighboring players, e.g. or_1 only knows or_0 and or_2's identity, while or_0 only knows or_1 and Alice' identity. If a circuit contains at least three onion routers, then, in principle, no single onion router should know both the sender and receivers identity. It will be assumed that sender Alice knows receiver Bob's identity. Receiver Bob on the other hand may not know Alice' identity.

An example of an onion routing anonymity network is shown in Figure 3.2, where all the circuits are of length 3, and each colored path denotes a circuit.

3.1.1 Replying to an Onion Message

If Alice wishes to get a reply from Bob, there are in principle three apparent ways to achieve this according to Goldschlag, Reed, and Syverson [9], which will be described in the following paragraphs. The first and simplest is that Alice tells Bob her identity and then Bob can make a circuit back to Alice. This method has, however, one major problem; Alice is no longer anonymous with respect to Bob. Alice will also no longer be anonymous with respect to the exit router either, i.e. the last encryption layer is removed by or_n to reveal the message and Alice' identity. Although Alice might trust Bob, there is no guaranty that or_n is trustworthy.

The next approach can be compared to that of including a return envelope when sending a letter/package in the mail. Before creating an onion from her message m, Alice selects return routers $or'_0, \cdots, or'_m \in$ N. These routers create a return path from Bob to Alice. This reply path may be the same as the forward circuit, but may also contain different onion routers all together. Alice creates a return onion r as follows: she encrypts her own address using the last reply onions key. The encrypted address gets appended to the last

[2]There exist systems where the last onion router in the circuit is not the exit router. This is called a leaky pipe topology [4]. This special case can be applied to the system here, but will not be explained or remain a focus for the remaining thesis.

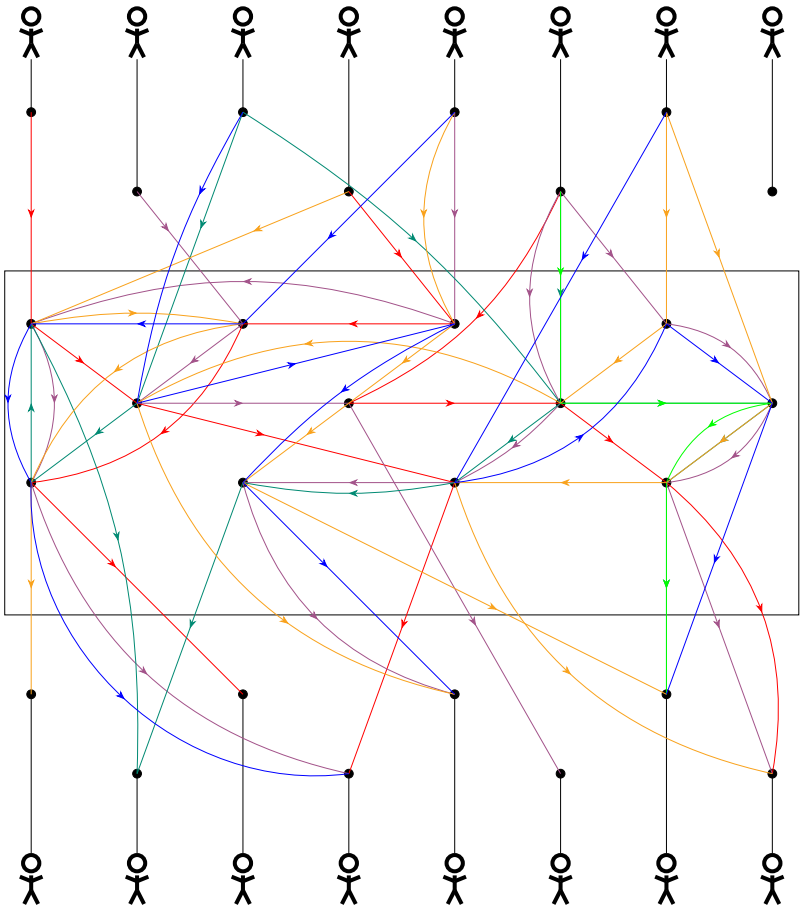


Figure 3.2: Example of an onion routing network with 8 users, 12 onion routers and 8 receivers. Each colored path represents a different circuit. Any adversary that cannot distinguish the circuits will then have a hard time following the circuit from sender to receiver.

onion routers identity before being encrypted with or'_{m-1}'s key. This continues until all the reply onions routers have an encryption layer on Alice' address. This creates an onion, r, as previously, the only difference being that the next destination is encrypted into the onion too.

Alice then appends her message m to r. She creates an onion from $m' = m||r$ as usual before sending it to or_0. At the final onion router in the circuit the message m' is revealed. or_n sends m to Bob as usual. If or_n receives a reply from Bob, it packs the reply in r before sending it to the next OR according to r's instructions. The reply then travels through the network much like a regular onion before arriving at Alice.

This method removes the anonymity breach that occurred in the previous method, but it limits the number of replies Bob can make. Since a reply onion is needed in order for Bob to reply, and a reply onion only can be used once, the number of replies Bob can

make between each message from Alice is limited to the number of reply onions Alice has appended.

The final return method works as follows. Whenever an onion router receives an onion that is going in return, it adds an encryption layer using a backward key[3] corresponding to the sender (creator) of the circuit, before sending the layered onion to the previous player of the circuit. When the message arrives at Alice it comes as a layered onion, with one encryption layer for each onion router in the circuit. Since the sender, Alice, knows all the decryption keys she can peel off all the encryption layers and get to the message from Bob. See Figure 3.3

The benefit of this method is that it neither limits the number of replies, nor does it reveal the senders identity. It does, however, require that the circuit is still running in order for replies to come through, something the previous two did not.

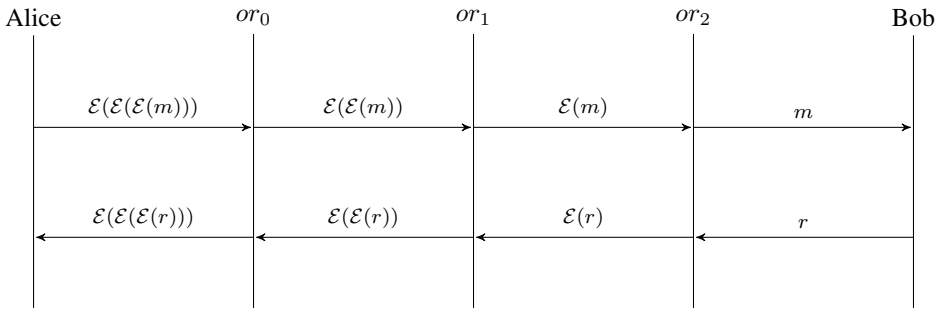


Figure 3.3: Alice communicating with Bob through a circuit consisting of or_0, or_1 and or_2. The upper arrows shows how the message m gets decrypted while passing along the circuit. The lower part shows how encryption layers gets added to the onion as the reply r travels back along the circuit.

3.2 Tor

This section will give an overview of Tor [4, 5]. Tor is one of the largest onion routing networks in the world with millions of users and thousands of onion routers [22].

In Tor, each onion router is equipped with a longterm private key for RSA-encryption, as well as a shortterm TLS-key, used to maintain the TLS-tunnel between all other onion routers in the network [4].

3.2.1 Circuit Construction

In order to communicate, the circuit has to be set up. A visualization of circuit creation can be seen in Figure 3.4.

According to Dingledine, Mathewson, and Syverson [4] the circuit construction works as follows:

[3] In the case of a shared symmetric key, the backward and forward keys are the same.

A sender, Alice, selects onion routers, $or_0, or_1 \cdots or_n \in$ N according to a selection algorithm. This selection algorithm is created such as to lower the probability that corrupted nodes become the entry or exit node of the circuit. Exactly how this is done, however, will not be covered in this thesis. Specially interested readers are referred to *Tor Path Specification* [3] for more information.

The sender first sends a create-message to or_0 with the first part of a the DH-handshake encrypted using the onion routers public key. The OR responds with the second part of the DH-handshake and a hash of the agreed upon symmetric key using the two parts required from DH. Now or_0 and the sender have a private symmetric communication channel.

To extend the circuit to the next onion router Alice computes a new, first part of DH-handshake, and encrypts it using the second onion routers public key. She appends the address of the second router to the encrypted DH-handshake and encrypts this again using the symmetric key she now shares with or_0. She then dispatches this onion to the first onion router. It decrypts the onion using its symmetric key and sends the message inside on to or_1. or_1 chooses a second half of the DH-handshake and together with a hash of the agreed upon key, sends it back to or_0. The entry router, or_0, encrypts the received onion using the symmetric key before sending it back to Alice.

Alice can now decrypt the message using the symmetric key shared with the first onion router to get access to the second part of the DH-handshake and thus have a shared symmetric key with or_1 too.

This process is repeated for or_2 through or_n to complete the circuit construction. In order to connect to the receiver the sender sends an onion along the circuit asking the exit router to create a TCP connection between itself and the desired receiver. It will be assumed that the message sent between exit routers and receivers do not reveal any information about the senders. This can easily be achieved by encrypting the messages before sending them.

Communication can then continue as described in the previous section.

3.2.2 Circuit Id's

Onions sent over Tor are sent in **cells** of 512 bytes[4] [5]. A cell consists of two parts: a 3 byte header and a payload, see Figure 3.5a. It is the payload that gets wrapped in encryption layers while the header remains in plain text.[5] The header is again split in two; a 2-byte **circuit id**, c_{id}, unique at every onion router along the circuit, and a 1-byte circuit command. Circuit commands will not be discussed, but denotes the type of cell. E.g. create new circuit (create), send onion along circuit (relay) etc.

During circuit creation, each onion router in turn receives a create cell. The create cell has an incoming circuit id as well as the first part of DH. The onion router who receives the cell selects an outgoing circuit id that is currently unused for any other circuit passing through the OR. In an internal database the onion router stores the incoming and outgoing circuit id pairs, together with the previous and next onion routers in the circuit, as well as the session key agreed upon with the sender. When a relay onion reaches an onion router,

[4] Some circuit commands allow cells of different length, but is not important for this thesis.

[5] Although, remember all communication along the circuit happen through TLS-tunnels, meaning the header is also encrypted.

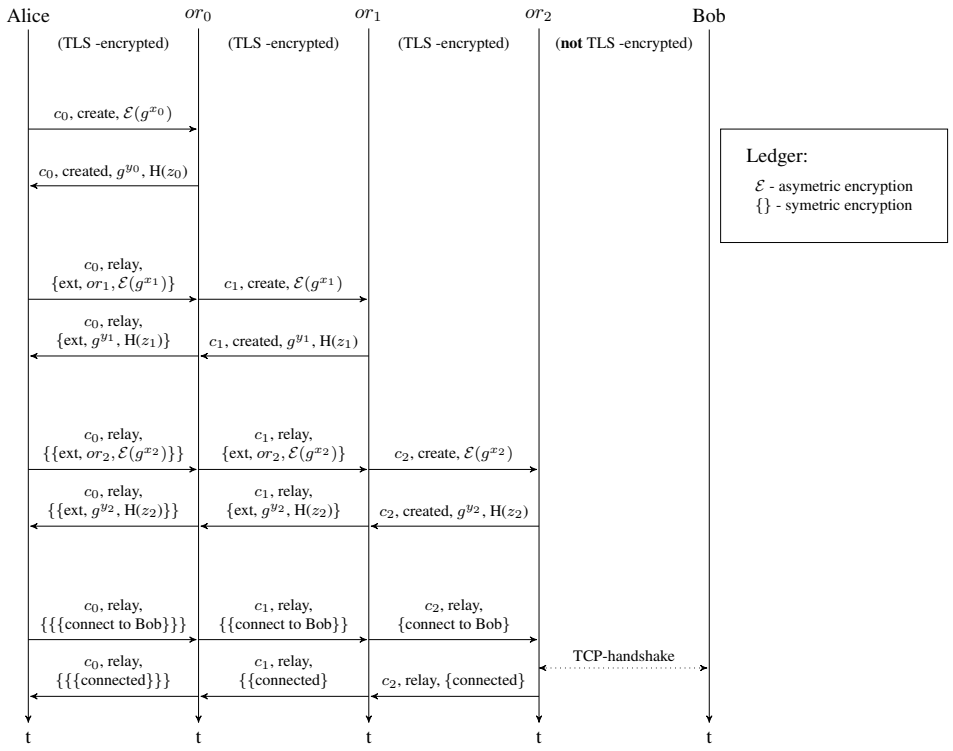


Figure 3.4: Constructing a Tor-circuit consisting of three onion routers or_0, or_1 and or_2. Alice first agrees upon a symmetric key with or_0. She then agrees on a symmetric key with or_1 by sending the first part of a DH-handshake as an onion through or_0. The same is done in order to achieve a shared symmetric key with or_2 as well. Finally, when all the symmetric keys are established, she creates a *connect to Bob*-onion that she sends along the circuit, which tells the last OR, or_2 to open a TCP connection to Bob.

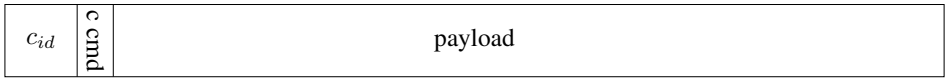
the OR looks up its incoming circuit id, changes it to the outgoing circuit id, peals off an encryption layer using the symmetric key, and sends it to the next onion router according to the database [4].[6]

Note that two circuits following the same path are still distinct due to the uniqueness of circuit id's.

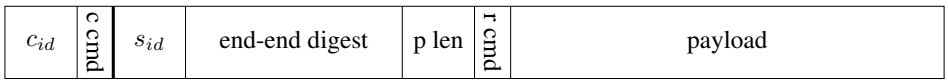
The circuit id will be a vital part in the definitions of anonymity goals given in Chapter 5. Any incoming circuit id is at the same time an outgoing circuit id for some other onion router. Whenever a circuit id corresponding to an onion router is mentioned it is the incoming circuit id that is meant. Given a player or and a circuit id c_{id} used by or on some circuit, i.e. onion router or with corresponding circuit id c_{id}. We write $or_{c_{id}}$ to denote the circuit through or corresponding to c_{id}.
Further we define

$$c_{id}(X) := \{x_{c_i} \mid x \in X,\ c_i \text{ incoming circuit id belonging to onion router } x, X \subseteq \mathrm{N}\}$$

[6]It will also pad the payload in order to keep the cell size a constant 512 bytes.



(a) Any cell consist of a 3 byte header and a payload. It is the payload that gets wrapped in encryption layers while the header remains in plain text. The header is again split in two; a 2-byte c_{id}, unique at every onion router along the circuit, and a 1-byte circuit command.



(b) Decrypted payload in a relay-cell: consist of a header, and a payload. The relay-header consist of a stream id, s_{id}. The stream id makes it possible to communicate with multiple recipients using the same circuit. Each connection has its unique stream id. The remaining bytes of the payload header are: end to end digest, it checks the integrity of the cell, if integrity is not intact the circuit gets torn down. Payload length tells the exact length of a message, then a message might have to be spread over multiple cells. Relay command is the final part of a relay header and indicates the type of message.

Figure 3.5: Illustration of a Tor cell. Ratios are not correct and only for visualization purposes.

to be the set of all onion routers in X with all their corresponding circuit id's. X is a subset of all onion routers in N. $c_{id}(\mathrm{P_N})$ will be assumed to be public knowledge.

3.2.3 Multiple Connections Along one Circuit

The decrypted payload in a relay-cell consists of another header, and a payload, see Figure 3.5b. The relay-header contains a **stream id** (s_{id}), which makes it possible to communicate with multiple recipients using the same circuit. Each connection has its unique stream id. The remaining bytes of the payload header are: a) end to end digest, used to check the integrity of the cell at the endpoints. If the integrity is not intact the circuit gets torn down. b) Payload length states the exact length of a message. A message might have to be spread over multiple cells. c) Relay command indicates the type of message, eg. create TCP, extend circuit etc.

A sender will by default reuse an already existing circuit for 10 minutes, when connecting to a new receiver [21]. If no such circuits exist, a new circuit will be created. Connections already made along a circuit will continue to communicate over the circuit even past the 10 minute period, but no more streams will be added. E.g. if a sender, Alice, wishes to communicate with a new receiver, Carol, she will first see if she has any already existing circuits created less than 10 minutes ago. If one exists she will connect to Carol through this, otherwise a new circuit will be created in order to establish a communication circuit with Carol. If, e.g. after 15 minutes, Alice wishes to connect to Dave she will have to create a new circuit, although she may continue to communicate with Carol along the already existing connection.

3.3 Relations between Players in an Onion Routing Network

When analyzing anonymity networks, an important aspect is whether or not two players lay on the same circuit, or whether a player lies on a specific circuit. A relation is introduced to do so unambiguously.

Definition 3.3.1. Two players $x, y \in \mathrm{P_N}$ are **related** if they are part of the same extended circuit.

We write $x \sim y$ to denote that x and y are related.

Remark. *Although we use $\sim$ as the relation symbol it is not an equivalence relation then transitivity does in general not hold. Even though x and y are on the same circuit and y and z are on a circuit, these two circuits are not necessarily the same, meaning x does not necessarily lie on a circuit with z.*

Definition 3.3.2. Two sequences of players $x_0 - \cdots - x_n$, $y_0 - \cdots - y_m$ in the anonymity network N are called **related** if both sequences lie on the same extended circuit.

We write $x_0 - \cdots - x_n \sim y_0 - \cdots - y_m$ to denote that $x_0 - \cdots - x_n$ and $y_0 - \cdots - y_m$ are related.

Definition 3.3.3. Player x is said to be **related to onion router r with respect to circuit id, $\mathbf{c_{id}}$**, if x lies on an extended circuit with r and the circuit has incoming circuit id c_{id} at r. Denote this relation by $x \sim r_{c_{id}}$.

It is easy to see that $\sim$ is both reflexive and symmetric. Any player is automatically related to itself, since they both lie on the same (spot on the) extend circuit. Furthermore, if x is in a circuit with y then y is also in a circuit with x.

If further development of the model is desirable these relations should be generalized to work for arbitrary anonymity networks.

Chapter 4

Network Simulation and Adversarial Power

We will now take a step back from onion routing and specify network simulation, independent of the type of anonymity network.

The objective of this section is to create adversarial powers that represents attack powers available in low-latency networks today.

A major difference between an adversary in an anonymity network setting vs. an adversary in a security setting is the amount of information they receive. For instance, Eve sitting in the security setting may see encrypted messages traveling back and forth between Alice and Bob. She may even witness a key exchange of some sort. The information Eve receives in that setting is very clear and straight forward. She sees everything with the exception of private calculations done by either Alice or Bob. Her challenge, in such a setting, is determining the information behind the communication she observes. In an anonymity network, however, it is not quite as simple. If Eve were to see the entire network and all the *public* information, the system might be considered broken [4]. Therefore Eve will only view parts of the network, meaning there is public information that Eve will not know. The challenge in an anonymity network is therefore not to determine the information inside the messages, but rather piecing together the information she observes to determine communication patters.

In addition to being able to observe traffic on parts of the network, Eve may even in certain cases alter or add her own traffic. In order to determine what Eve sees on different parts of the network, as well as when she may alter traffic, we need a way of classifying the power Eve has over specific players.

This leads to the following definitions.

Definition 4.0.1. A player $x \in \mathrm{P_N}$, is called **corrupt** by an adversary $\mathcal{A}$ if $\mathcal{A}$ has complete power over its functionality.
Let $\mathrm{X_N} \subseteq \mathrm{P_N}$. We denote $\mathrm{Cor}_{\mathcal{A}}(\mathrm{X_N})$ as the subset of $\mathrm{X_N}$ corrupted by $\mathcal{A}$. Furthermore, $\mathrm{Hon}_{\mathcal{A}}(\mathrm{X_N})$ denotes the set of honest users. I.e. $\mathrm{Hon}_{\mathcal{A}}(\mathrm{X_N}) \cup \mathrm{Cor}_{\mathcal{A}}(\mathrm{X_N}) = \mathrm{X_N}$ and $\mathrm{Hon}_{\mathcal{A}}(\mathrm{X_N}) \cap \mathrm{Cor}_{\mathcal{A}}(\mathrm{X_N}) = \emptyset$. The subscript is omitted when $\mathcal{A}$ is clear from context.

Definition 4.0.2. A player, $x \in \mathrm{P_N}$, is called **observable** by an adversary $\mathcal{A}$ if the traffic to and from x can be viewed by $\mathcal{A}$.
Let $\mathrm{X_N} \subseteq \mathrm{P_N}$. We denote $\mathrm{Obs}_{\mathcal{A}}(\mathrm{X_N})$ as the subset of $\mathrm{X_N}$ observable by $\mathcal{A}$. The subscript is omitted when $\mathcal{A}$ is clear from context.

Remark. *A player that is corrupted by $\mathcal{A}$ is also observable by $\mathcal{A}$.*
I.e. $Cor_{\mathcal{A}}(X_N) \subseteq Obs_{\mathcal{A}}(X_N)$.

If x is observable by $\mathcal{A}$ then by definition all traffic to and from x can be observed by the adversary $\mathcal{A}$. The underlying transport method between all players in an anonymity network is based on TCP or UDP, independent of whether there is a TLS-tunnel on top or not [10]. TCP/UDP headers contain both source and destination port which are unencrypted [11]. This means that an adversary who can see the traffic to and from x will also know about its next and previous players. This gives an adversary, that can view x, information about players it may otherwise not have had any knowledge about.

The discussion above gives the rational for the following definition.

Definition 4.0.3. A player $x \in \mathrm{P_N}$ is called **semi-observable** for an adversary $\mathcal{A}$ if x is observable or if there exists a player $x' \in \mathrm{Obs(P_N)}$ such that x', x are adjacent players in N.
Let $\mathrm{X_N} \subseteq \mathrm{P_N}$. We denote $\text{Semi-Obs}_{\mathcal{A}}(\mathrm{X_N})$ as the subset of $\mathrm{X_N}$ semi-observable by $\mathcal{A}$. The subscript is omitted when $\mathcal{A}$ is clear from context.

Remark. *Note that $Obs(X_N) \subseteq Semi\text{-}Obs(X_N)$.*

The example network shown in Figure 4.1 shows the onion network from Figure 3.2 from an adversarial point of view, where the red nodes are corrupted while the blue nodes are observable players. Observe that the node in the top-right corner, although not observable is known to the adversary due to its correspondence with a corrupted player. The sender next to it, however, remains invisible since it does not communicate through any observable entry nodes.

Further more, in the center bottom part of the image notice how there is a black connection between two black nodes at the same time as there is a dashed gray edge. This comes from the fact that the sender of the black edge is corrupted. The adversary therefore knows all players in its circuit, but since neither the receiver nor the exit router is observable by the adversary any other connections made between these two players remain invisible to the adversary.

4.1 Network Simulation

In order to have useful and interesting reductions later on, it would be desirable to give the adversary as much power as possible, without letting the anonymity notions be trivially broken. An initial idea could, therefore, be to let the adversary decide the traffic itself. This, however, turns out to be too much power. The adversary can then simply stop all traffic along the network, with the exception of the connection it is interested in. From this it is no hard task to deduce who is communicating with whom by simply observing whether or not the traffic is as expected.

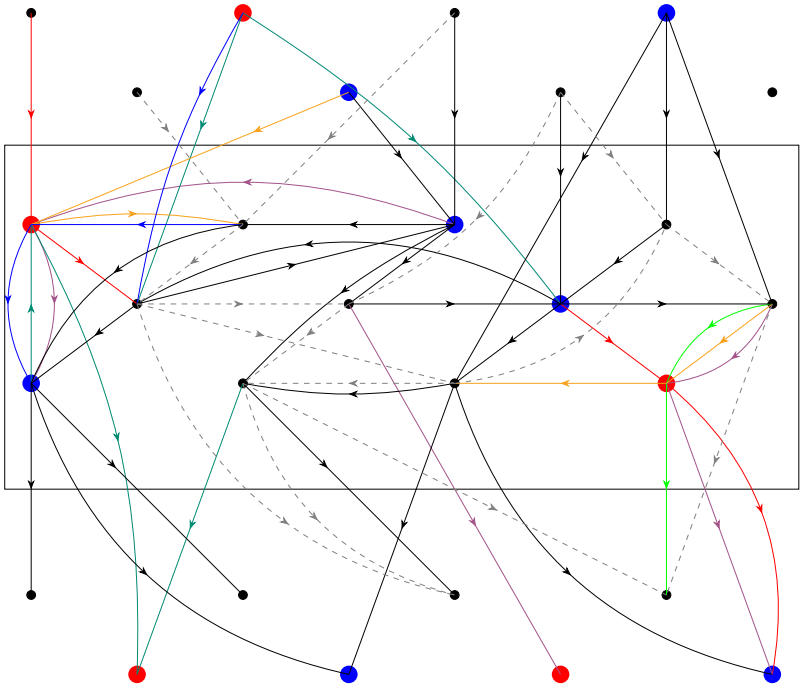


Figure 4.1: An onion-network from an adversarial point of view, where blue nodes are non-corrupted observable players and red nodes are corrupted players. Black nodes with edges connected to them are semi-observable players. Black edges in the graph symbolizes that the adversary can observe the traffic on that edge but does not recognize which circuit it belongs to. Dashed gray edges are connections not visible for the adversary. There may be multiple circuits communicating over the same black or gray edge. Colored edges are as previously specific circuits.

The solution is, therefore, to have an impartial third-party simulator, *Sim*, simulate the traffic over a network N. *Sim* simulates the traffic according to some internal instructions, meaning that different simulators will result in different network behaviors. The simulator runs independently of whether or not there is an adversary present.

When an adversary $\mathcal{A}$ wants to analyze the network, it informs the simulator and gets a **partial network** of N, denoted [N], in return. [N] corresponds to the part of N that is semi-observable by the adversary.

The simulator will then simulate the usage of the network and sends information to $\mathcal{A}$ about what is viewable/observable by the adversary, $\mathcal{A}$. For traffic concerning corrupted players of $\mathcal{A}$, $\mathcal{A}$ may choose to alter, stop or create traffic. The adversary can do this by sending the alterations it wishes to make back to the simulator. The simulator will then return any network action observable by $\mathcal{A}$, continuously as they happen, to $\mathcal{A}$. Any information that passes through a corrupted player temporarily pauses all traffic that is simulated on N until $\mathcal{A}$ responds back to *Sim* with whether it wishes to alter the specific traffic. If no change is desirable $\mathcal{A}$ returns $\perp$.

Although on a different technical level than the remaining thesis, the formal definition of a simulator will include a Turing machine. This is done in order to have a well defined

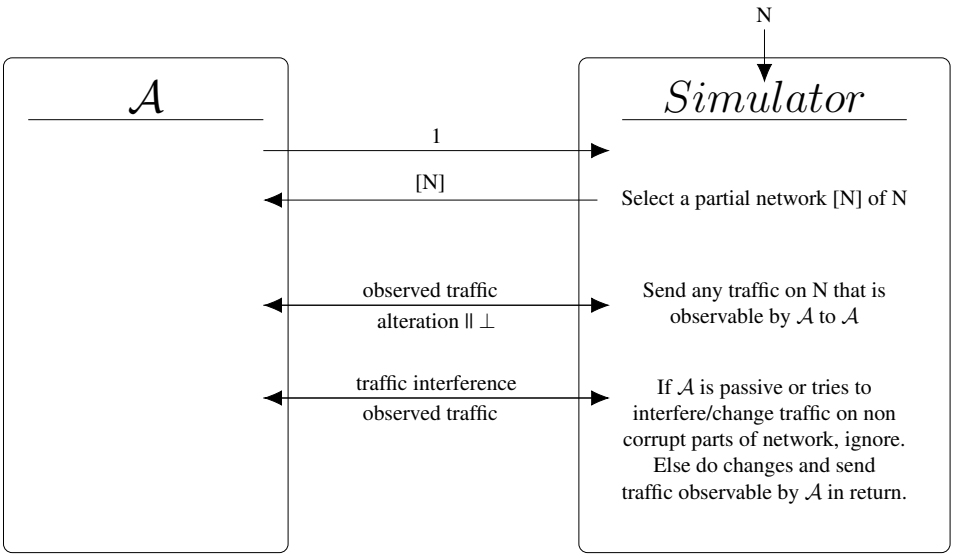


Figure 4.2: A simulator simulating the network N. When $\mathcal{A}$ wishes to enter the network he sends 1 to the simulator. The simulator selects a partial network [N], that will be semi-observable for $\mathcal{A}$, which it returns to $\mathcal{A}$. Whenever traffic happens in the network that is semi-observable to $\mathcal{A}$, *Simulator* sends the information to $\mathcal{A}$. When $\mathcal{A}$ wishes to interfere on the network he sends the wanted changes to *Simulator*.

technical process that explains its inner workings.

When later working with a simulator we will, however, try to think of it as depicted in Figure 4.2.

Definition 4.1.1. A simulator, *Sim*, is a probabilistic (non-deterministic) Turing machine, with four infinite tapes.[1] The first tape takes the network description, N, as input. The second takes traffic action request from any corresponding party. The third tape is semi-observable output for the same corresponding party. The final tape is a regular work tape. The corresponding party can only see the second and third tape, and only write on the second.
When *Sim* receives 1 on the second tape, *Sim* non-deterministically decide on [N] based on its current state and its underlying automata description, and writes this onto tape two. It then continues to simulate as before, now checking tape two after every step to see if any traffic alterations are requested. If the request does not correspond with allowed change it is simply ignored. *Sim* outputs all traffic that is observable for the corresponding party according to [N] onto tape three. On certain outputs it enters a "waiting" state and will not continue simulation before a response is written on tape two.

From the definition of the simulator it is clear that the network behavior is largely based on the underlying automata of *Sim*. It is, therefore, clear that the network properties may largely vary depending on the type of simulator being used. For instance a simulator

[1] It is trivial to change a Turing machine with one infinite tape to one with four infinite tapes. Simply say each of the four tapes get every fourth slot on the original tape.

that only sends a message along every connection once every second, will be immune to timing attacks. The exact underlying automata of *Sim* will not be discussed here.

The simulator definition is as mentioned earlier a very technical definition, and the simulator will not be thought of as a Turing machine. It will rather be though of as a black box that simulates the network traffic, as shown in Figure 4.2.

4.2 Passive and Active Adversaries

Adversarial powers can in general be divided into two different types: passive and active.

A **passive adversary** will only view the traffic on parts of the network. An **active adversary** has the same capabilities as a passive adversary but will in addition have influential powers over certain players, and can thus alter traffic on certain parts of the network. I.e. a key difference between active and passive adversaries lies in the fact that the former has corrupt players, and can, therefore, influence the network, whereas the passive only has observable players.

It is obvious that a passive adversary always can be created from an active adversary, of the same power, by simply not changing the traffic, or returning $\perp$ to *Sim* when a response is required. Creating an active adversary from a passive is, on the other hand, not always possible. Not answering or giving wrong information to the active adversary may lead to a malfunction of the original active adversary.

Another benefit of using the simulator model is that an adversary may be inserted into the network at any point in time. In other words, it does not have to observe the creation of the network, but may be put to analyze and attack a network that is already in existence. This has the added benefit that when creating an active adversary $\mathcal{A}$ from another active adversary $\mathcal{B}$, $\mathcal{A}$ may use $\mathcal{B}$ multiple times. This would not have been possible if an adversary had to see the network from the start. $\mathcal{A}$ could potentially *record* the network creation and run this multiple times through $\mathcal{B}$, but if $\mathcal{B}$ is active it might decide to alter the traffic differently on different runs. $\mathcal{A}$ has no way of answering such request as the situation $\mathcal{B}$ is reacting to may be long gone.

We say that an adversary $\mathcal{A}$ can be **reduced** to an adversary $\mathcal{B}$ if one can create adversary $\mathcal{A}$ using $\mathcal{B}$.

4.3 General Low-Latency Adversarial Powers

A simulation can be adapted to any unspecified anonymity network by only informing the adversary of behaviors related to the endpoints, sender and receivers. In this section the attack powers will be introduced independent of the anonymity network. Then in the next section they will be specified to work for Tor-specific onion networks.

Any oracles introduced later in this chapter will be integrated with the simulator. They are only introduced as stand alone oracles in order to achieve cleaner definitions.

The standard approach would be to give the adversary necessary information to do the conclusions itself. This method, however, makes anonymity notions introduced in Chapter 6 much simpler to relate.

All the oracles are equipped with a success probability ρ indicating how often the oracle returns the correct answer to an inquiry made by an adversary. The probability ρ mimics the real world situation where only a percentage of the conclusions made by an adversary about the network are correct. In order to keep the model simple the oracle only has one success probability. However, if further development of the model were to happen, we suggest dividing the success probability of an oracle in two; the success of true positive and true negative results.

It is worth pointing out that moving the model to an idealized situation may lead to loss of information which again may lead to weaker adversaries.

The following adversarial powers are taken from Dingledine, Mathewson, and Syverson [5] and formalized to fit the structure of the network model.

Due to time constraints not all their proposed attack-models have been included in this thesis, and is left for future work.

4.3.1 Timing Attack

Timing attack is considered a universal power for any low-latency anonymity network adversary, and will be an underlying power for all other attack powers introduced in this thesis.

In an actual anonymity network timing attack works by correlating the time a message leaves one observable endpoint with the time it enters another [16]. If these times always vary by a consistent amount then there is a certain probability that they communicate.

Timing attacks will in this model be introduced as a probabilistic oracle that on two observable endpoints returns whether or not they communicate, with a certain success. By using an oracle instead of letting the adversary analyze the traffic itself we remove the inconvenience of having to incorporate time/timestamps into our model.

Definition 4.3.1 (Timing attack - endpoints)**.** Let $\mathcal{O}$ be an oracle that on input (x, y), two endpoints, outputs $\mathcal{O}(x, y) = b \in \{0, 1\}$. Furthermore, let E denote the event that x communicates with y and $x, y \in \text{Obv}(\text{S}_\text{N} \cup \text{R}_\text{N})$. We say that an adversary $\mathcal{A}$ has ρ**-endpoint timing attack power (ETA)** if $\mathcal{A}$ has access to $\mathcal{O}$ and

$$Succ(\mathcal{O}) = P((b = 1) \cap \text{E}) + P((b = 0) \cap \neg\text{E}) \geq \rho.$$

As remarked above, all low-latency attack models are vulnerable against timing attacks, it will, therefore, be assumed that all the attack models described in this section and the next include (endpoint) timing attack power. (E)TA will in these cases be called the **implicit** attack power and any additional attack power will be called **explicit**.

4.3.2 Chosen Sender Attack

Chosen sender attack illustrates the setting where an adversary tries convincing the human sender to give up their correspondent. The probability that the sender cooperates is given by ρ.

It may seem like too much power to give to the adversary, since it automatically links the sender and receiver, and in a general low-latency network this might be the case, since

only senders and receivers are specified. However, in specific networks where the other players N are specified it not as simple. If one recalls the anonymity goals mentioned in the introduction non of them necessarily asked for senders and receivers to be linked in order to break the system. They mostly involve determining the sender/receiver of specific communications. Hence chosen sender attack might actually not automatically break the challenge.

Definition 4.3.2 (Chosen sender attack)**.** Let $\mathcal{O}$ be an oracle that on input x a sender returns $y = \mathcal{O}(x)$, a receiver to $\mathcal{A}$. Let E denote the event that x communicates with y. We say that an adversary $\mathcal{A}$ has ρ**-chosen sender attack power (CEA)** if $\mathcal{A}$ has access to $\mathcal{O}$ and $Succ(\mathcal{O}) = P(\mathrm{E}) \geq \rho$.

Note here that giving x as input to CEA oracle $\mathcal{O}$ multiple times may result in various outputs, i.e. receivers from different circuits or different receivers from the same circuit.

Chosen receiver attack, on the other hand, is in general not possible. Then the sender is usually unknown to the receiver.

4.3.3 Network Altering Attacks

It would also be natural to assume that the adversary may infiltrate the network further dependent on the traffic it observes.

However, an adversary that can view the entire network, and all its traffic, will with large probability be able to link communicating players through timing attack. This is one of the major draw backs of low-latency networks [16]. Such a network is considered broken and is, therefore, of no interest to us. It can be viewed as the equivalent of the adversary knowing the private key in a cryptosystem. Therefore, it is clearly necessary to have some restriction on how many players an adversary may infiltrate.

There are generally two ways an adversary may infiltrate the network. It may either position it self in such a way that it observes the traffic of a chosen player, or it may corrupt the player altogether. As with the previous attacks there is always an uncertainty to whether or not the attack actually succeeded.

Definition 4.3.3 (n-endpoint observation attack)**.** An adversary $\mathcal{A}$ has ρ-n-**endpoint observation attack** power if $\mathcal{A}$ has access to an oracle, $\mathcal{O}$, n times, that on input $x \in \mathrm{R_N} \cup \mathrm{S_N}$, notifies the simulator that x is observable by $\mathcal{A}$ with probability at least ρ. If $\mathcal{A}$ can successfully observe x, $\mathcal{O}$ returns $b = 1$, else returns $b = 0$.

Definition 4.3.4 (n-endpoint corruption attack)**.** An adversary $\mathcal{A}$ has ρ-n-**endpoint corruption attack** power if $\mathcal{A}$ has access to an oracle, $\mathcal{O}$, n times, that on input $x \in \mathrm{R_N} \cup \mathrm{S_N}$ notifies the simulator that x is corrupted by $\mathcal{A}$ with probability at least ρ. If corruption is successful $\mathcal{O}$ returns $b = 1$, else returns $b = 0$.

The limit n may be restricted even further after the anonymity notions have been introduced.

4.3.4 Relating the General Attack-Powers

It is clear from the definitions above that certain adversarial models should be more powerful than others. Which gives us the following inclusions.

Theorem 4.3.1. *n-endpoint corruption attack $\implies$ n-endpoint observation attack*

Proof. Follows from the fact that every corrupted player is observable. □

Theorem 4.3.2. *If an adversary can use its explicit attack power at most n times, then n-endpoint corruption attack $\implies$ Chosen sender attack. We denote this as:*

$$n\text{-endpoint corruption attack} \stackrel{n}{\Rightarrow} \text{Chosen sender attack.}$$

Proof. A CEA oracle $\mathcal{O}_{CEA}$ will on input x with probability at least ρ return y where x communicates with y. ECA replicates this power by corrupting x. Since x is a user it by assumption knows all its communication partners. ECA can then just pick one and output that. □

4.4 Tor Specific Adversarial Powers

In a Tor specific network the role of the network players are specified, and a natural consequence of this is to let an adversary use these players to its advantage.

4.4.1 Timing Attack

For timing attack it is reasonable to extend the end point definition to include all observable players. The definition can, however, be extended even further.

An adversary can distinguish different circuits going through a corrupt player due to the circuit id's. It is, therefore, natural to introduce timing attack on specific circuits as well, e.g. is player x relate to circuit $r_{c_{id}}$?

Two onions that enter an observable player r from different players r', r'' are clearly not on the same circuit, therefore, asking whether x and $r - r'$ or x and $r - r''$ are timing correlated can reveal more information than simply asking about x and r. This can again be extended to cover observable-paths in the network of arbitrary length. Using this observation it is natural to extend timing attack to work on such observable paths.

The adversary also knows about the existence of semi-observable players through TCP headers. It is, therefore, reasonable to allow timing attack on paths that start or end in semi-observable players.

This discussion gives rise to the following Tor specific timing attack. However, before it can be properly defined, certain preliminaries need to be made.

Let B_0 be the set of all sequences of players observable by an adversary $\mathcal{A}$. I.e.

$$\mathrm{B}_0 = \{x_0 - x_1 - \cdots - x_n \,|\, x_i \in \mathrm{Obs}_{\mathcal{A}}(\mathrm{P_N}), n \geq 0\}$$

Let B_1, B_2 and B_3, respectively, be the set of all sequences of observable players *starting*, *ending* and *starting and ending* with semi-observable players.

Let B_4 be the union of the sets B_0 through B_3. I.e.

$$\begin{aligned}
\mathrm{B}_1 &= \{x - \vec{y} \,|\, x \in \text{Semi-Obs}_{\mathcal{A}}(\mathrm{P_N}),\, \vec{y} \in \mathrm{B}_0\} \\
\mathrm{B}_2 &= \{\vec{y} - x \,|\, x \in \text{Semi-Obs}_{\mathcal{A}}(\mathrm{P_N}),\, \vec{y} \in \mathrm{B}_0\} \\
\mathrm{B}_3 &= \{x - \vec{z} \,|\, x \in \text{Semi-Obs}_{\mathcal{A}}(\mathrm{P_N}),\, \vec{z} \in \mathrm{B}_2\} \\
\mathrm{B}_4 &= \mathrm{B}_0 \cup \mathrm{B}_1 \cup \mathrm{B}_2 \cup \mathrm{B}_3.
\end{aligned}$$

Now for the actual timing attack definition.

Definition 4.4.1 (Timing attack)**.** Let $\mathrm{B} = c_{id}(\text{Cor}_{\mathcal{A}}(\mathrm{P_N})) \cup B_4$. Let $\mathcal{O}$ be an oracle that on input $(\mathcal{X}, \mathcal{Y})$, outputs $\mathcal{O} = b \in \{0, 1\}$. Let E denote the event that $\mathcal{X} \sim \mathcal{Y}$ and $\mathcal{X}, \mathcal{Y} \in \mathrm{B}$. We say that an adversary $\mathcal{A}$ has ρ**-timing attack power (TA)** if $\mathcal{A}$ has access $\mathcal{O}$ and $Succ(\mathcal{O}) = P((b = 1) \cap E) + P((b = 0) \cap \neg E) \geq \rho$.

Note that the adversary can only do timing attacks on specific circuits when the player the circuit belongs to is corrupted. Hence, although the adversary knows all the circuit id's a player uses, they will not be useful for timing attacks.

It will in this section, as in the previous, be assumed that any attack model has TA as its implicit attack power.

It is tempting to incorporate a chosen player attack here. An obstacle arises, however, from the fact that players that are not senders do not know other players on the circuit other than their neighbors. Hence chosen player attack is not a realistic attack-power.

4.4.2 Network Altering Attacks

Since network players are defined in a Tor specific network it would make sense to let the adversary infiltrate onion routers as it desires.

However, as in the previous section, the adversary may not freely infiltrate, since a low-latency anonymity network consisting only of corrupted players is considered broken.

Definition 4.4.2 (n-observation attack)**.** An adversary $\mathcal{A}$ has ρ-n**-observation attack** power if $\mathcal{A}$ has access to an oracle, $\mathcal{O}$, n times, that on input $x \in \mathrm{P}$ notifies the simulator that x is observable by $\mathcal{A}$ with probability at least ρ. If $\mathcal{A}$ can successfully observe x, $\mathcal{O}$ returns $b = 1$, else returns $b = 0$.

Definition 4.4.3 (n-corruption attack)**.** An adversary $\mathcal{A}$ has ρ-n**-corruption attack** power if $\mathcal{A}$ has access to an oracle, $\mathcal{O}$, n times, that on input $x \in \mathrm{P}$ notifies the simulator that x is corrupted by $\mathcal{A}$ with probability at least ρ. If corruption is successful, $\mathcal{O}$ returns $b = 1$, else returns $b = 0$.

4.5 Relating Attack-Powers

It is clear that the Tor specific adversaries all contain the same powers as the corresponding endpoint adversary, this gives rise to the following inclusion.

Theorem 4.5.1. *X-attack* $\implies$ *endpoint X-attack,*
X$\in \{$*timing, n-corruption, n-observability*$\}$

Proof. With the exception of chosen sender attack that does not exist in a Tor-specific setting all the other endpoint attacks are restrictions of their Tor-specific counterparts. □

As with the general attack models, any adversary that may corrupt n players, can also observe n players.

Theorem 4.5.2. n*-corruption attack* $\implies$ n*-observation attack*

Proof. Follows from definitions, and the fact that every corrupt player is observable. □

The attack powers introduced in this chapter, and the theorems proving certain relations between them, gives rise to the following diagram Figure 4.3.

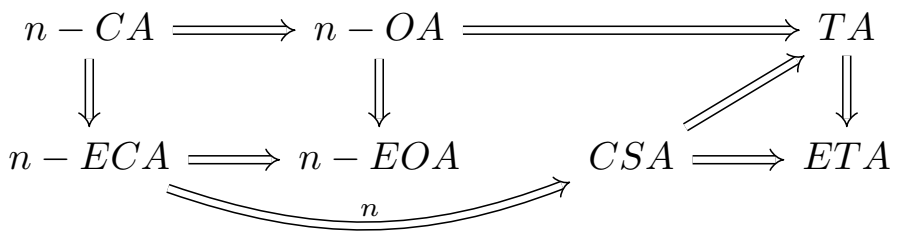


Figure 4.3: The diagram shows the different attack models introduced in this section and how they relate. An implication from attack model X to Y indicates that X includes the powers of Y. The powers introduced are (endpoint) corruption attack ((E)CA) which lets an adversary corrupt players at will; (Endpoint) observation attack ((E)OA) that makes players observable; Chosen sender attack (CSA), where an adversary can convince a sender to reveal its communication partner; and (E)TA, (endpoint) timing attack correlates whether observable players lie on the same circuit.

Chapter 5

Anonymity Goals

One of the main goals of an anonymity network is hiding the sender and receiver identities. Given access to part of the anonymity network, an adversary should not be able to identify the sender/receiver of a specific communication path unless absolutely trivial. It might appear strange to talk about receiver anonymity, as they might not be human, however, these terms are used for anonymity networks in general and we will, therefore, apply the same terminology.

In this chapter onion routing-specific anonymity goals will be introduced. These goals will be formulated with minimal information about the adversarial powers. Only the adversaries relation to the anonymity network will be of any relevance.

In Section 1.1 it became clear that anonymity is not a universal property but defined with respect to a specified entity, hence the adversaries may not be entirely removed.

The works of Pfitzmann and Hansen [17] contain characteristics of what an anonymity network should be able to withstand. One such characteristic is **sender anonymity**; it shall not be possible to determine which player is the sender x of a specific circuit thorough onion router r. I.e. given an onion router r and a corresponding circuit id c_{id} the sender x should be indistinguishable form the other senders. If, however, r is corrupted and $x - r$ forms part of an extended circuit, then this goal is trivially broken. Similarly, if players $z_0, z_1, \cdots, z_n$ are corrupt, where z_n has outgoing circuit id c_{id}, and $x - z_0 - \cdots - z_n - r$ is part of an extended circuit, the question is again trivial.

An adversary should not be said to have broken an anonymity goal when the challenge is trivially broken through corruption. A solution could be to simply prevent the adversary to have corrupt players. However, forcing sender anonymity by preventing an adversary from corrupting players altogether, would drastically reduce the adversary's powers. It is also desirable to have the adversarial powers be as realistic as possible, and since anyone can create an onion router [20] such a solution would only lessen potential results.

We would therefore like to say that a sender has sender anonymity as long as it is not directly connected to the onion router in question through corrupted OR's. This gives rise to the following preliminary definition before the anonymity goals can be properly defined.

Definition 5.0.1. Two players $x, y \in \mathrm{P_N}$ are **directly connected** in N for an adversary $\mathcal{A}$ if there exist players $r_0, r_1, \cdots, r_n \in$ Cor(N), $n \in \mathbb{N}_0$, such that $x - r_1 - \cdots - r_n - y$ is a subsequence of an extended circuit.

The definitions of Sender (Receiver) anonymity (SA (RA)) and Unlinkability in the following sections are taken form the work of Pfitzmann and Hansen [17] and adapted to fit the current model.

5.1 Anonymity-Specific Anonymity Goals

With the proper terminology available, sender and receiver anonymity can now be defined.

Both sender and receiver versions of all goals will be defined where applicable. However, only the sender versions will be analyzed.

Definition 5.1.1 (Sender (Receiver) anonymity (SA (RA)))**.** Let $\mathcal{A}$ be an adversary that on input (r, c_{id}), where $r \in$ N and c_{id} corresponding circuit id, returns $x \in \mathrm{S_N}$ $(x \in \mathrm{R_N})$.
Let E denote the event that $x \sim r_{c_{id}}$ given that x, r are not directly connected for $\mathcal{A}$.
Define the success of $\mathcal{A}$ as $Succ(\mathcal{A}) = P(\mathrm{E})$. The network N is then defined to have ϵ**-sender (receiver) anonymity** with respect to $\mathcal{A}$ if for all input pairs (r, c_{id}):

$$Adv(\mathcal{A}) = \frac{|\mathrm{S_N}|}{|\mathrm{S_N}| - 1} \left| Succ(\mathcal{A}) - \frac{1}{|\mathrm{S_N}|} \right| \leq \epsilon$$

$$\left(Adv(\mathcal{A}) = \frac{|\mathrm{R_N}|}{|\mathrm{R_N}| - 1} \left| Succ(\mathcal{A}) - \frac{1}{|\mathrm{R_N}|} \right| \leq \epsilon \right)$$

Note that an adversary may always be lucky and guess the correct sender x with probability $\frac{1}{|\mathrm{S_N}|}$. The success of an adversary should, therefore, always be greater than or equal to $\frac{1}{|\mathrm{S_N}|}$. Assuming an adversary is correct less than $\frac{1}{|\mathrm{S_N}|}$ of the time, another adversary may use this to their advantage by always selecting another sender than the one the first selected. This second adversary will then have a success probability higher than $\frac{1}{|\mathrm{S_N}|}$.

Intuitively it would be wrong to call the success of an adversary its advantage, as this would imply an edge the adversary does not have. Simply guessing should not indicate a benefit. Therefore, an adversary's success rate is corrected for guessing. In order let the advantage range between 0 and 1, the corrected success is then normalized by $\frac{|\mathrm{S_N}|}{|\mathrm{S_N}|-1}$.

In cryptography it is currently unclear whether decisional Diffie-Hellmann (DDH) and computational Diffie-Hellman (CDH) are equivalent, even though intuitively DDH should be easier [13]. It would, therefore, be interesting to compare an adversary who has to *compute* the sender (receiver) of a circuit to an adversary that has to *decide* which of two senders (receivers) is an endpoint of a specified circuit.

Definition 5.1.2 (Decisional sender (receiver) anonymity (DSA (DRA)))**.** Let $\mathcal{A}$ be an adversary that on input (x_0, x_1, r, c_{id}), where $x_0, x_1 \in \mathrm{S_N}$ $(x_0, x_1 \in \mathrm{R_N})$, $r \in$N and c_{id} a corresponding circuit id, returns $b \in \{0, 1\}$.
Let E denote the event that $x_b \sim r_{c_{id}}$ given that $x_{1-b} \not\sim r_{c_{id}}$ and the pairs x_0, r and x_1, r are not directly connected for $\mathcal{A}$.

Define the success of $\mathcal{A}$ as $Succ(\mathcal{A}) = P(\mathrm{E})$. The network N is then defined to have **ϵ-decisional sender (receiver) anonymity** with respect to $\mathcal{A}$ if for all input tuples (x_0, x_1, r, c_{id}):

$$Adv(\mathcal{A}) = 2\left|Succ(\mathcal{A}) - \frac{1}{2}\right| \leq \epsilon$$

Another interesting challenge to give the adversary would be circuit decisional sender (receiver) anonymity. The adversary has to determine which of two circuits a sender (receiver) is communicating over.

Definition 5.1.3 (Circuit decisional sender (receiver) anonymity (CDSA (CDRA)))**.** Let $\mathcal{A}$ be an adversary that on input (x, r_0, c_0, r_1, c_1), where $x \in \mathrm{S_N}$ ($x \in \mathrm{R_N}$), $r_0, r_1 \in \mathrm{N}^2$ with respectively corresponding circuit id's c_0, c_1, returns $b \in \{0, 1\}$.
Let E denote the event that $x \sim r_{bc_b}$ given that $x \not\sim r_{(1-b)c_{1-b}}$ and the pairs x, r_0 and x, r_1 are not directly connected.
Define the success of $\mathcal{A}$ as $Succ(\mathcal{A}) = P(\mathrm{E})$. The network N is defined to have **ϵ-circuit decisional sender (receiver) anonymity** with respect to $\mathcal{A}$ if for all input tuples (x, r_0, c_0, r_1, c_1):

$$Adv(\mathcal{A}) = 2\left|Succ(\mathcal{A}) - \frac{1}{2}\right| \leq \epsilon$$

Note the difference between DSA (DRA) and CDSA (CDRA) is that the former ask an adversary to differentiate between a circuits sender and another random sender. The latter, on the other hand, asks an adversary to determine which of two circuits a specific sender is communicating over.

Again, for both DSA (DRA) and CDSA (CDRA) the advantage is normalized and regulated as not to indicate any unjustifiable advantage.

5.1.1 Relating Anonymity-Specific Anonymity Goals

Now that we have defined the anonymity-specific anonymity goals it would be interesting to see how these goals relate. I.e. if whenever network N is X anonymous is N also Y anonymous? This is the same as saying if there exists an adversary that breaks Y on N, then there exists an adversary that breaks X on N. We denote this reduction by $X \implies Y$.

In the remaining thesis, reductions will be proven contrapositively using adversaries.

Theorem 5.1.1. *Let N be a network. If N is ϵ-DSA anonymous then N is ϵ-SA anonymous.*

Proof. Let $\mathcal{A}'$ be an adversary that breaks SA on N with advantage greater than ϵ. The goal is to create an adversary $\mathcal{A}$ that breaks DSA on N with $Adv(\mathcal{A}) > \epsilon$.

When $\mathcal{A}$ is given its challenge x_0, x_1, r, c_{id}, the idea is to simply run $\mathcal{A}'$ on the challenge circuit. If $\mathcal{A}'$ outputs either of the challenge senders, x_0, x_1, $\mathcal{A}$ returns the index corresponding to that sender. Else $\mathcal{A}$ outputs a number uniformly at random from $\{0, 1\}$.

Define $\mathcal{A}$ as follows:

$\mathcal{A}(x_0, x_1, r, c_{id})$:

$x' \leftarrow \mathcal{A}'(r, c_{id})$
if $x' = x_{b'}$ where $b' \in \{0,1\}$
then
  return b'
else
  return $b' \xleftarrow{r} \{0,1\}$
end if

Let E denote the event E given in the definition of DSA. Let C, W denote, respectively, the event that $\mathcal{A}'$ selects the right x_i, and the event that $\mathcal{A}'$ selects the wrong x_i from $\{x_0, x_1\}$. Note that $Succ(\mathcal{A}') = P(C)$. Then the success of $\mathcal{A}$ is given as follows:

$$
\begin{aligned}
Succ(\mathcal{A}) &= P(E) \\
&= P(E \mid C)P(C) + P(E \mid W)P(W) + P(E \mid \neg C \cap \neg W)P(\neg C \cap \neg W) \\
&= 1 \cdot P(C) + 0 \cdot P(W) + \frac{1}{2}P(\neg C \cap \neg W) = Succ(\mathcal{A}') + \frac{1}{2}P(\neg C \cap \neg W) \\
&= Succ(\mathcal{A}') + \frac{1}{2}P(\neg(W \cup C)) = Succ(\mathcal{A}') + \frac{1}{2}(1 - P(W \cup C)) \\
&= Succ(\mathcal{A}') + \frac{1}{2}(1 - (P(W) + P(C) - P(W \cap C))) \\
&= Succ(\mathcal{A}') + \frac{1}{2}(1 - P(W) - Succ(\mathcal{A}')) = \frac{1}{2}(Succ(\mathcal{A}') + 1 - P(W))
\end{aligned}
$$

The step between the last two lines in the equation above comes from the fact that $\mathcal{A}'$ only outputs one sender. This means events C and W can never occur at the same time, i.e. $P(C \cap W) = 0$.

Although $\mathcal{A}'$ might have a predisposition to choose certain wrong senders, given a specific router and circuit id pair, this disposition cancels itself out, assuming the other challenge x_i is randomly picked and outside $\mathcal{A}'$s knowledge. The probability that $\mathcal{A}'$ selects the wrong user of x_0 and x_1 without having any knowledge of them, is the same as $\mathcal{A}'$ selecting a user and then the simulator picking the wrong challenge user afterwards. The simulator, knows nothing about $\mathcal{A}'$ and must thus pick the not correct x_i uniformly at random. Hence the probability that *Sim* chooses the sender $\mathcal{A}'$ selected is $\frac{1-Succ(\mathcal{A}')}{|\mathrm{S_N}|}$. Hence, $P(W) = \frac{1-Succ(\mathcal{A}')}{|\mathrm{S_N}|-1}$ and we get:

$$
\begin{aligned}
Adv(\mathcal{A}) &= 2\left|\frac{1}{2}\left(Succ(\mathcal{A}') + 1 - \frac{1 - Succ(\mathcal{A}')}{|\mathrm{S_N}| - 1}\right) - \frac{1}{2}\right| \\
&= \left|Succ(\mathcal{A}') - \frac{1 - Succ(\mathcal{A}')}{|\mathrm{S_N}| - 1}\right| \\
&= \left|\frac{(|\mathrm{S_N}| - 1)Succ(\mathcal{A}') + Succ(\mathcal{A}') - 1}{|\mathrm{S_N}| - 1}\right| \\
&= \frac{|\mathrm{S_N}|}{|\mathrm{S_N}| - 1}\left|Succ(\mathcal{A}') - \frac{1}{|\mathrm{S_N}|}\right| = Adv(\mathcal{A}') > \epsilon
\end{aligned}
$$

□

The contrapositive is not as clear, since the SA-adversary will not known which two senders x_0 and x_1 to give to the DSA-adversary.

However, if the DSA-adversary is perfect, i.e. has advantage 1, then a reduction is possible: This is due to the fact that if DSA-adversary is perfect it is clear that when the correct sender x is one of x_0 and x_1, DSA-adversary will output x. Which means that every other sender x' looses at least once against x. If we run DSA-adversary on all pairs of senders, there is one and only one sender that will win every run, namely the correct sender.

Theorem 5.1.2. *Let N be a network. Let $\mathcal{A}'$ be an adversary that perfectly breaks DSA on N. Then there exists an adversary $\mathcal{A}$ that perfectly breaks SA on N.*

Proof. Let $\mathcal{A}'$ be an adversary that breaks DSA of N with advantage 1. The goal is to create an adversary $\mathcal{A}$ that breaks SA with $Adv(\mathcal{A}) = 1$.

Define $\mathcal{A}$ as follows:

$\mathcal{A}(r, c_{id}, \mathbf{S_N})$:

Fix an enumeration of $\mathrm{S_N}$.
winner $\leftarrow (0, 0, \cdots, 0) \in \mathbb{Z}^{|\mathrm{S_N}|}$
for all $(x_i, x_j) \in \mathrm{S_N^2}$ where $i < j$
do
  $k \leftarrow \mathcal{A}'(x_i, x_j, r, c_{id})$
  $k \leftarrow \begin{cases} i, & \text{if } k = 0. \\ j, & \text{otherwise.} \end{cases}$
  $\mathrm{winner}_k \leftarrow \mathrm{winner}_k + 1$
end for
$b' \leftarrow$ argmax(winner)
return $x_{b'}$

Every sender in $\mathrm{S_N}$ is run $|\mathrm{S_N}| - 1$ times through $\mathcal{A}'$, hence we get that $\mathrm{winner}_i \in \{0, \cdots, |\mathrm{S_N}| - 1\}$.

Since $\mathcal{A}'$ has success probability 1, every time either x_i or x_j is the correct sender x, $\mathcal{A}'$ returns the correct index. Hence the only sender that wins every time is x (since all the other senders loose at least once, against x). Therefore, $Succ(\mathcal{A}) = 1 \implies Adv(\mathcal{A}) = 1$. □

What is interesting here is that in a world of perfect adversaries, SA and DSA are equivalent.

SA reduced to DSA, however, needs to run the DSA-adversary multiple times in order to reach a conclusion. For the converse only one run of the SA-adversary is needed. Simulation vise there is no problem concerning this. There might, however, be a problem when the explicit power of the multiple run adversary is limited. The constructed adversary may not have enough runs available to solve the challenge.

To conclude, SA and DSA are equivalent in a world of perfect adversaries, when no limits are enforced. However, the need for multiple runs indicate that DSA has an imagined advantage in the perfect adversary world.

Just as with the DSA to SA reduction, if an adversary can determine the sender of a circuit, it should be no harder to determine which of two circuits a sender is using. The adversary can simply determine the sender of both circuits and then output the circuit that has the desired player as sender.

Theorem 5.1.3. *Let N be a network. If N is ϵ-CDSA anonymous then N is ϵ-SA anonymous.*

Proof. Let $\mathcal{A}'$ be an adversary that breaks SA of N with advantage greater than ϵ. The goal is to create an adversary $\mathcal{A}$ that breaks CDSA with $Adv(\mathcal{A}) > \epsilon$.

Define $\mathcal{A}$ as follows:

$\mathcal{A}(x, (r_0, c_{id_0}), (r_1, c_{id_1}))$:

---

if $r_{0c_0} \notin c_{id}(\{r_0\})$ **then**
  return $b \leftarrow 1$
else if $r_{1c_1} \notin c_{id}(\{r_1\})$ **then**
  return $b \leftarrow 0$
end if
# By this point it is known that both circuit pairs are valid. Technically not needed since the input has to be valid for the adversary to win.
$y_0 \leftarrow \mathcal{A}'(r_0, c_{id_0})$
$y_1 \leftarrow \mathcal{A}'(r_1, c_{id_1})$
if ($y_b = x$ and $y_{1-b} \neq x$ where $b \in \{0, 1\}$) **then**
  return b
else
  return $b \xleftarrow{r} \{0, 1\}$
end if

Let E denote the event E given in the definition of CDSA.
Let C denote the event that $\mathcal{A}'$ outputs x on the correct circuit.
Let W denote the event that $\mathcal{A}'$ outputs x on the wrong circuit.

Since C and W come from different runs of $\mathcal{A}'$, which are independent, C and W are independent. Further $P(C) = Succ(\mathcal{A}')$ and similar to discussion given in the proof of Theorem 5.1.1 $P(W) = \frac{1-Succ(\mathcal{A}')}{\mathrm{S_N}-1}$.

$$
\begin{aligned}
Succ(\mathcal{A}) &= P(E) \\
&= P(E \mid C \cap W)P(C \cap W) + P(E \mid \neg C \cap W)P(\neg C \cap W) \\
&\quad + P(E \mid C \cap \neg W)P(C \cap \neg W) + P(E \mid \neg C \cap \neg W)P(\neg C \cap \neg W) \\
&= \frac{1}{2}P(C \cap W) + 0P(\neg C \cap W) + 1P(C \cap \neg W) + \frac{1}{2}P(\neg C \cap \neg W) \\
&= \frac{1}{2}P(C)P(W) + P(C)(1 - P(W)) + \frac{1}{2}(1 - P(C))(1 - P(W)) \\
&= \frac{1}{2}P(C)P(W) + P(C) - P(C)P(W) \\
&\quad + \frac{1}{2}\left(1 - P(W) - P(C) + P(C)P(W)\right) \\
&= \frac{1}{2}(1 + P(C) - P(W)) = \frac{1}{2}\left(1 + Succ(\mathcal{A}') - \left(\frac{1 - Succ(\mathcal{A}')}{|\mathrm{S_N}| - 1}\right)\right) \\
&= \frac{|\mathrm{S_N}| - 1 + |\mathrm{S_N}|Succ(\mathcal{A}') - Succ(\mathcal{A}') - 1 + Succ(\mathcal{A}')}{2(|\mathrm{S_N}| - 1)} \\
&= \frac{|\mathrm{S_N}| - 2 + |\mathrm{S_N}|Succ(\mathcal{A}')}{2(|\mathrm{S_N}| - 1)}
\end{aligned}
$$

Which means

$$
\begin{aligned}
Adv(\mathcal{A}) &= 2\left|Succ(\mathcal{A}) - \frac{1}{2}\right| = 2\left|\frac{|\mathrm{S_N}| - 2 + |\mathrm{S_N}|Succ(\mathcal{A}')}{2(|\mathrm{S_N}| - 1)} - \frac{1}{2}\right| \\
&= \left|\frac{|\mathrm{S_N}|Succ(\mathcal{A}') - 1}{|\mathrm{S_N}| - 1}\right| = \frac{|\mathrm{S_N}|}{|\mathrm{S_N}| - 1}\left|Succ(\mathcal{A}') - \frac{1}{\mathrm{S_N}}\right| = Adv(\mathcal{A}') > \epsilon
\end{aligned}
$$

□

It would be nice to do a proof of the converse theorem using a method similar to that used when reducing SA to DSA. There are, however, a number of obstacles concerning said method, even when using perfect adversaries. The first and foremost being that CDSA determines which circuit a sender is using not who the sender is.

In the $SA \implies DSA$ reduction the perfect-adversary used the fact that there was exactly one sender that would win every single one of its challenges in order to solve the challenge. The same would apply for SA to CDSA reduction if it was not for the fact that a sender may have multiple circuits. If an CDSA adversary is given a sender and two of

its circuits, there is no guarantee that it will select the circuit the adversary is interested in. Therefore, there is no guarantee that the correct circuit with the correct user will win the most.

5.2 Unlinkability

In addition to determining the endpoint anonymity, another interesting question to ask about an anonymity network is whether or not two players are related. If an adversary is not able to determine this the two items are referred to as **unlinkable**.

Unlinkability does not, per se, refer to anonymity, but is a useful tool for determining how much information an adversary knows about an anonymity network.

Formally we define unlinkability as follows:

Definition 5.2.1 (Unlinkability)**.** Let $\mathcal{A}$ be an adversary that on input $x, y \in \mathrm{P_N}$ returns $b \in \{0, 1\}$. Let E_0 denote the event that $x \sim y$ given that x, y are not directly related. Let E:= $(\mathrm{E}_0 \cap b = 1) \cup (\neg \mathrm{E}_0 \cap b = 0)$.
Two players x, y are called ϵ**-unlinkable** if $Succ(\mathcal{A}) = P(\mathrm{E})$ and

$$Adv(\mathcal{A}) = 2\left|Succ(\mathcal{A}) - \frac{1}{2}\right| \leq \epsilon.$$

Furthermore a network N is called ϵ**-unlinkable** if all pairs of not directly connected players in $\mathrm{P_N}$ are ϵ-unlinkable.

It might also be interesting to know how unlinkable one specific player r is. I.e. Determine whether all players not directly connected to r are unlinkable to r. One can define this setting by restricting the definition of unlinkability to only ask whether a player lies on a circuit with a fixed player r. This gives rise to the following definition.

Definition 5.2.2 (Unlinkability through r)**.** Let $r \in \mathrm{P_N}$. Let $\mathcal{A}$ be an adversary that on input $x \in \mathrm{P_N}$ returns $b \in \{0, 1\}$. Let E_0 denote the event that $x \sim r$ given that x, r are not directly related. Let E:= $(\mathrm{E}_0 \cap b = 1) \cup (\neg \mathrm{E}_0 \cap b = 0)$.
A player x are called ϵ**-unlinkable through** r if $Succ(\mathcal{A}) = P(\mathrm{E})$ and

$$Adv(\mathcal{A}) = 2\left|Succ(\mathcal{A}) - \frac{1}{2}\right| \leq \epsilon.$$

Furthermore a network N is called ϵ**-unlinkable through** r if all players in $\mathrm{P_N}$ not directly connected to r are ϵ-unlinkable.

Clearly an adversary $\mathcal{A}$ that breaks unlinkability will also break unlinkability through r, e.g. if $\mathcal{A}$ is asked to determine whether player x is ϵ-unlinkable through r, it can simply check if x, r are ϵ-unlinkable. This discussion gives rise to the following theorem.

Theorem 5.2.1. *ϵ-unlinkability through r $\implies$ ϵ-unlinkability.*

Proof. Let N be an anonymity network. Let $\mathcal{A}'$ be an adversary that breaks unlinkability of N with advantage greater than ϵ. The goal is to create an adversary $\mathcal{A}_r$ that breaks unlinkability of N through r with $Adv(\mathcal{A}_r) > \epsilon$.

Define $\mathcal{A}_r$ as follows:

$\mathcal{A}_r(x)$:
$b' \leftarrow \mathcal{A}'(x, r)$
return b'

Let E denote the event E described in the definition of unlinkability through r.
Let C denote the event that $\mathcal{A}'(x, r)$ outputs correct b'.

$$\begin{aligned} Succ(\mathcal{A}_r) = P(\mathrm{E}) &= P(\mathrm{E} \mid C)P(C) + P(\mathrm{E} \mid \neg C)P(\neg C) \\ &= 1P(C) + 0P(\neg C) = Succ(\mathcal{A}'). \end{aligned}$$

Hence:

$$Adv(\mathcal{A}_r) = 2\left|Succ(\mathcal{A}_r) - \frac{1}{2}\right| = Adv(\mathcal{A}') > \epsilon.$$

□

The converse is not as clear. An unlinkable through r adversary $\mathcal{A}_r$ cannot conclusively determine whether or not two players, x, y, are unlinkable if neither of them is r. It might conclude that $x \sim r$ and $r \sim y$. However, since $\sim$ is not a transitive relation that does not imply $x \sim y$.

However, if an adversary $\mathcal{A}$ had access to an unlinkability through r adversary for every $r \in \mathrm{P_N}$, the reduction becomes trivial again. Hence:

Theorem 5.2.2. *ϵ-Unlinkability $\implies$ ϵ-Unlinkability through r $\forall r \in P_N$.*

Proof. Let N be an anonymity network. Let $\mathcal{A}_r$ be an adversary that breaks unlinkability through r of N with advantage greater than ϵ. The goal is to create an adversary $\mathcal{A}$ that breaks unlinkability of N with $Adv(\mathcal{A}_r) > \epsilon$.

Define $\mathcal{A}$ as follows:

$\mathcal{A}(x, y)$:
$b' \leftarrow \mathcal{A}_y(x)$
return b'

Let E denote the event E described in the definition of unlinkability.
Let C denote the event that $\mathcal{A}_y(x)$ outputs correct b'.

$$Succ(\mathcal{A}) = P(\mathrm{E}) = P(\mathrm{E} \mid C)P(C) + P(\mathrm{E} \mid \neg C)P(\neg C)$$
$$= 1P(C) + 0P(\neg C) = Succ(\mathcal{A}_r).$$

Hence:

$$Adv(\mathcal{A}) = 2\left|Succ(\mathcal{A}) - \frac{1}{2}\right| = Adv(\mathcal{A}_r) > \epsilon.$$

□

From the two previous theorems it is clear that unlinkability and unlinkability through r for all $r \in \mathrm{P_N}$ are equivalent. This is true for any adversary, and we will only include unlinkability in later reductions.

5.3 Anonymity and Unlinkability Relations

From the theorems in the two previous sections we get the following diagram of how the anonymity goals relate. The dashed arrow indicates a reduction only in the perfect adversarial setting.

Figure 5.1: Relations among ϵ-anonymity goals. The implications indicate how the anonymity goals relate, the dashed arrow indicates a reduction only in the world of perfect adversaries.

Chapter 6

Anonymity Notions

Anonymity notions are anonymity goals combined with attack powers. A network N that is anonymous with respect to some anonymity notion A-B, is an anonymity network that will remain A-anonymous against adversaries of type B. In other words an anonymity notion indicates a general level of robustness that a network with that notion must have.

Before the attack powers and anonymity goals can be combined there are certain precautions that need to be made with regards to the attack powers. Ideally an adversary should be allowed to pick its own challenge. However, the potential need to run an adversary multiple times in a reduction will be problematic when there is no guarantee the original adversary asks for the same challenge twice. The question of whether or not one should let the adversary select its own challenge turns, therefore, into a question of preference, since either option has its drawbacks: An adversary that select its own challenge might not break the anonymity notion on the challenge it is given, there might, however, exist a valid challenge that it could break. On the other hand, an adversary that does not select its own challenge can be reducible to another adversary $\mathcal{A}'$ even if $\mathcal{A}'$ is allowed multiple runs. The latter would not be *possible* if the adversaries were to select their own challenges, then there is no guarantee that $\mathcal{A}'$ ask for the same challenge multiple times if run twice. This would make a second run of $\mathcal{A}'$ potentially useless.

In this thesis we view the ability to use adversaries multiple times in reductions greater than having the adversary select its own challenge. This is due to the fact that the simulator selects the challenge independent of the adversary and its powers, which is, after all, something the simulator has no knowledge of. The only information it can base its challenge on is [N], hence it can be assumed *Sim* selects its challenge among valid challenges uniformly at random. This means that the simulator will in general give the adversary challenges of average difficulty.

Since the anonymity goals defined in Chapter 5 are Tor-specific, Tor-specific attack models will be used where applicable. Similarly CSA will have timing attack as its implicit power. This gives a total of $5 \cdot 4 = 20$ anonymity notions. However, due to the equivalence of Unlinkability and Unlinkability through r, as proven in the previous chapter, only the former is discussed further, giving a total of 16 anonymity notions.

The goal for the remainder of this chapter is to present the anonymity notion and introduce some relations among them.

6.1 Anonymity Games

Anonymity notions can be viewed as games between an adversary and the simulator, where the adversary has access to an oracle of the prescribed power. Just as before, the oracle is incorporated into the simulator. The games precede as follows:

An adversary informs the simulator of its presence, where it gets a partial network [N] in return. When the adversary is ready to play, it informs the simulator who returns a challenge. At some point the adversary has to output what it believes the answer of the challenge to be. During the entire process the adversary may use its attack powers according to its own desires. The adversary is said to win if it breaks the challenge, and thus the anonymity notion.

The formal definitions, however, are virtually identical to those given in Chapter 5, only with an added line concerning the adversary's power. There is also an added requirement that the adversaries have to break the anonymity notions in polynomial time in order for it to be considered broken. This added requirement is to prevent the adversary from brute-forcing its way to a solution where such is applicable.

Definition 6.1.1 (Sender anonymity -atk)**.** Let $\mathcal{A}$ be an adversary with attack power atk $\in$ {TA, CSA, n-OA, n-CA} that, on input (r, c_{id}), where $r \in$ N and c_{id} corresponding circuit id, returns $x \in \mathrm{S_N}$.
Let E denote the event that $x \sim r_{c_{id}}$ given that x, r are not directly connected for $\mathcal{A}$.
Define the success of $\mathcal{A}$ as $Succ(\mathcal{A}) = P(\mathrm{E})$. The network N is defined to be ϵ**-SA-atk** anonymous if $\mathcal{A}$ runs in polynomial time, and for all input pairs (r, c_{id}) $\mathcal{A}$ has advantage:

$$Adv(\mathcal{A}) = \frac{|\mathrm{S_N}|}{|\mathrm{S_N}| - 1} \left| Succ(\mathcal{A}) - \frac{1}{|\mathrm{S_N}|} \right| \leq \epsilon.$$

A game version of this definition can be seen in Figure 6.1a.

Definition 6.1.2 (Decisional sender anonymity -atk)**.** Let $\mathcal{A}$ be an adversary with attack power atk $\in$ {TA, CSA, n-OA, n-CA} that, on input, (x_0, x_1, r, c_{id}) where $x_0, x_1 \in \mathrm{S_N}$, $r \in$N and c_{id} a corresponding circuit id, returns $b \in \{0, 1\}$.
Let E denote the event that $x_b \sim r_{c_{id}}$ given that $x_{1-b} \not\sim r_{c_{id}}$ and x_0, r and x_1, r are not directly connected for $\mathcal{A}$.
Define the success of $\mathcal{A}$ as $Succ(\mathcal{A}) = P(\mathrm{E})$. The network N is defined to be ϵ**-DSA-atk** anonymous if $\mathcal{A}$ runs in polynomial time, and for all input tuples (x_0, x_1, r, c_{id}) $\mathcal{A}$ has advantage:

$$Adv(\mathcal{A}) = 2 \left| Succ(\mathcal{A}) - \frac{1}{2} \right| \leq \epsilon.$$

A game version of this definition can be seen in Figure 6.1b.

Definition 6.1.3 (Circuit decisional sender anonymity -atk)**.** Let $\mathcal{A}$ be an adversary with attack power atk $\in$ {TA, CSA, n-OA, n-CA} that, on input (x, r_0, c_0, r_1, c_1), where $x \in \mathrm{S_N}$, $r_0, r_1 \in \mathrm{N}^2$ and c_0, c_1 are circuit id's, returns $b \in \{0, 1\}$.
Let E denote the event that $x \sim r_{bc_b}$ given that $x \not\sim r_{(1-b)c_{1-b}}$ and x, r_0 and x, r_1 are not directly connected for $\mathcal{A}$.

Define the success of $\mathcal{A}$ as $Succ(\mathcal{A}) = P(\mathrm{E})$. The network N is defined be ϵ**-CDSA-atk** anonymous if $\mathcal{A}$ runs in polynomial time, and for all input tuples (x, r_0, c_0, r_1, c_1) $\mathcal{A}$ has advantage:

$$Adv(\mathcal{A}) = 2\left|Succ(\mathcal{A}) - \frac{1}{2}\right| \leq \epsilon$$

A game version of this definition can be seen in Figure 6.1c.

Definition 6.1.4 (Unlinkability -atk)**.** Let $\mathcal{A}$ be an adversary with attack power atk $\in \{$TA, CSA, n-OA, n-CA$\}$ that, on input, $x, y \in \mathrm{P_N}$ returns $b \in \{0, 1\}$.
Let E_0 denote the event that $x \sim y$ given that x, y are not directly related.
Let E:= $(\mathrm{E}_0 \cap b = 1) \cup (\neg \mathrm{E}_0 \cap b = 0)$.
Define the success of $\mathcal{A}$ as $Succ(\mathcal{A}) = P(\mathrm{E})$. The network N is considered to be ϵ**-unlinkable-atk** anonymous if $\mathcal{A}$ runs in polynomial time, and for all not directly related pairs of players x, y $\mathcal{A}$ has advantage:

$$Adv(\mathcal{A}) = 2\left|Succ(\mathcal{A}) - \frac{1}{2}\right| \leq \epsilon$$

A game version of this definition can be seen in Figure 6.1d.

6.2 Relating anonymity notions

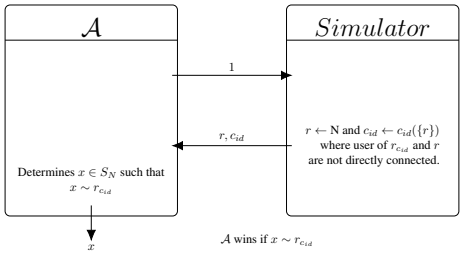
From the previous section it is clear that the reductions done in Chapter 4 and Chapter 5 are highly applicable to anonymity notions as well.

If an adversary $\mathcal{A}$ contains all the powers of another adversary $\mathcal{B}$ it is obvious that if $\mathcal{B}$ can break an anonymity goal A, so can $\mathcal{A}$. This translates to saying if network N is A-$\mathcal{A}$ anonymous then N is also A-$\mathcal{B}$ anonymous.

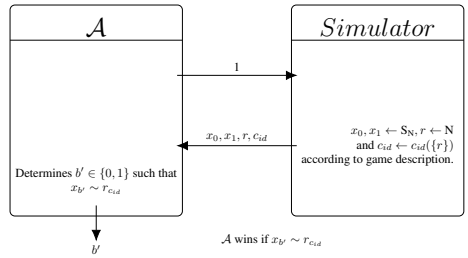
Similarly, from the reductions in Chapter 5 if an anonymity goal can be reduced to another anonymity goal independent of the adversary's attack power, then clearly by increasing the adversary's abilities this does not change.

Hence, we get the diagram, as shown in Figure 6.2, of reductions between anonymity notions where the horizontal arrows are adversarial power dependent, and the vertical implications are the reductions between anonymity goals. The reduction from SA to DSA, is not included, since it was only proven for a perfect adversary.

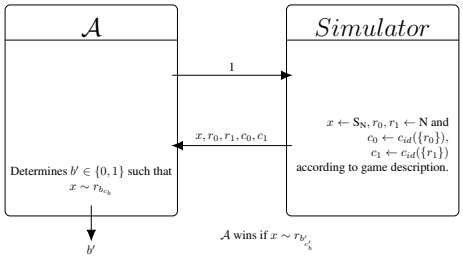
Note how the arrows from CDSA-n-CA and CDSA-n-OA to, respectively, SA-n-CA and SA-n-OA has a limit of maximum one half of the available explicit power uses. This comes from the fact that the reduction used two runs of the SA-adversary in order to create an CDSA-adversary. Therefore, CDSA-n-CA and CDSA-n-OA can only comply with half the requests due to its limit of explicit power.



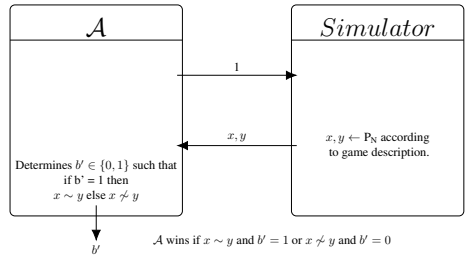
(a) **Game SA-atk:** Let N be a network. Let $\mathcal{A}$ be an adversary that has access to an oracle $\mathcal{O}$ of type atk $\in$ {TA, CSA, n-OA, n-CA}. $\mathcal{A}$ asks for a challenge by sending 1 to *Simulator*. *Simulator* then selects a router r and a circuit id, c_{id} corresponding to r and returns (r, c_{id}) to $\mathcal{A}$. $\mathcal{A}$ can consult the oracle both before and after the challenge is given. At some point $\mathcal{A}$ outputs an endpoint x'. $\mathcal{A}$ wins if $x' \sim r_{c_{id}}$.



(b) **Game DSA-atk:** Let N be a network. Let $\mathcal{A}$ be an adversary that has access to an oracle $\mathcal{O}$ of type atk $\in$ {TA, CSA, n-OA, n-CA}. $\mathcal{A}$ asks for a challenge by sending 1 to *Simulator*. *Simulator* then selects an onion router r, circuit id, c_{id}, corresponding to r and two senders x_0, x_1, $x_0 \neq x_1$, where $x_b \sim r_{c_{id}}, b \in \{0, 1\}$, returns it to $\mathcal{A}$. $\mathcal{A}$ can consult the oracle both before and after the challenge is given. At some point $\mathcal{A}$ outputs $b' \in \{0, 1\}$. $\mathcal{A}$ wins if $b' = b$.



(c) **Game CDSA-atk:** Let N be a network. Let $\mathcal{A}$ be an adversary that has access to an oracle $\mathcal{O}$ of type atk $\in$ {TA, CSA, n-OA, n-CA}. $\mathcal{A}$ asks for a challenge by sending 1 to *Simulator*. *Simulator* then selects sender $x \in$ S$_N$, $r_0, r_1 \in$ N with corresponding circuit id's c_0, c_1 such that $x \sim r_{b_{c_b}}$ and $x \not\sim r_{1-b_{c_{1-b}}}$, $b \in \{0, 1\}$. *Simulator* returns (x, r_0, c_0, r_1, c_1). $\mathcal{A}$ can consult the oracle both before and after the challenge is given. At some point $\mathcal{A}$ outputs $b' \in \{0, 1\}$. $\mathcal{A}$ wins if $b = b'$.



(d) **Game Unlink-atk:** Let N be a network. Let $\mathcal{A}$ be an adversary that has access to an oracle $\mathcal{O}$ of type atk $\in$ {TA, CSA, n-OA, n-CA}. $\mathcal{A}$ asks for a challenge by sending 1 to *Simulator*. *Simulator* then selects $b \in \{0, 1\}$ uniformly at random. If $b = 1$, select $x, y \in$ P$_N$ such that $x \sim y$ else select $x, y \in$ P$_N$ such that $x \not\sim y$. For either b, x can not be directly connected to y. *Simulator* returns x, y to $\mathcal{A}$. At some point $\mathcal{A}$ outputs $b' \in \{0, 1\}$. $\mathcal{A}$ wins if $b = b'$.

Figure 6.1: Games description of anonymity notions for an anonymity network N.

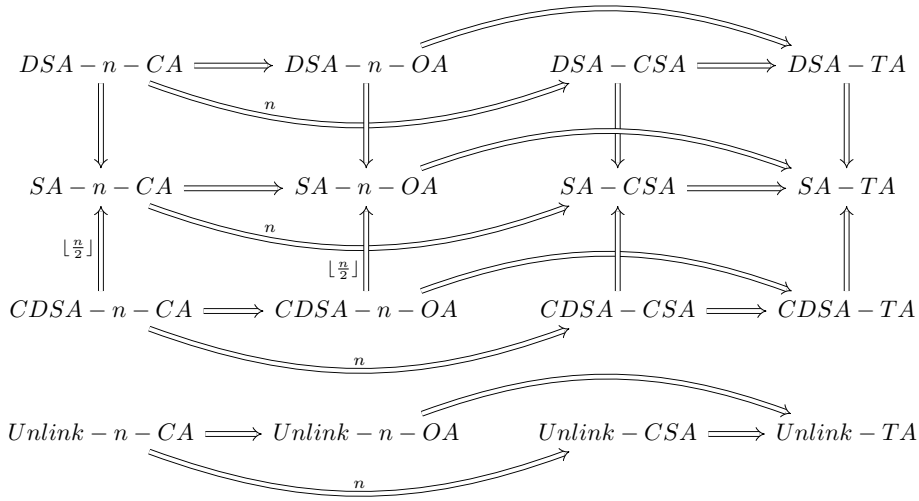


Figure 6.2: Relation among the different anonymity notions presented during this thesis. Where the horizontal reductions are between attack powers, and the vertical between anonymity goals.

Chapter 7

Applications

This chapter will apply the anonymity network model to a Tor-like network as well as another anonymity network that is not based on onion routing, and compare how the two different networks compare according to the anonymity notions.

Recall that in both the definition of anonymity goals and the definition of anonymity notions there is a requirement that the challenge players may not be directly related. Given an adversary with the power atk = n-CA, the adversary cannot corrupt players in such a way that the challenge players become directly connected. If they do, the challenge turns void and the adversary looses automatically.

Independent of the anonymity network, if an adversary $\mathcal{A}$ is given a challenge where non of the challenge players are semi-observable, $\mathcal{A}$ has no possible way of solving the challenge other than guessing. This comes from the fact that asking an adversary whether the traffic travels this or that way, when nothing can be seen, is essentially the same as simply tossing a coin and asking the adversary to determine the outcome. For DSA and CSA, with three challenge players, it is also insufficient to have only one semi-observable player, since determining the traffic flow can not be done by only observing one player.

If, however, enough challenge players are appropriately corrupt or observable, then the success of an adversary will be greater or equal to the success of the timing oracle. This comes form the simple fact that an adversary will always have access to a timing oracle, and can, therefore, simply ask about the connections.

How successful the timing oracle is, on the other hand, varies dependent on the way the network is simulated, i.e. the type of anonymity.

7.1 General Onion Routing

In an anonymity network such as Tor, where timing attack has a *high* success probability[1], it is clear from the discussion above that the network altering attacks n-OA and n-CA need to be limited even further.

[1] What the success of timing attack is seems to vary dependent on how many players the adversary can observe [12, 16].

If $n \geq 2$, no low-latency onion routing network will be CDSA-n-CA, DSA-n-CA or Unlink-n-CA anonymous. In the DSA situation, this comes from the fact that if an adversary with n-CA powers is given challenge (x_0, x_1, r, c_{id}) it can corrupt the onion router as well as one of the two possible related players. The adversary can then use the implicit timing attack power on the two corrupted players. If the TA-oracle returns *yes*, it outputs the index corresponding to the corrupted sender, else it returns the non-corrupted sender. A similar tactic can be applied to the anonymity goals CDSA and Unlink. It is, therefore, reasonable to say that $n = 1$, in n-CA, when paired with the anonymity goals DSA, CDSA and Unlinkability.

For SA it is not as simple since the adversary, $\mathcal{A}$, has to identify the user itself. It can start by corrupting the challenge router r to get information about the circuit in question. However, without following the circuit, which would essentially break the challenge, there is no straight forward way of finding the correct sender. If, however, $n \geq |\mathrm{N} \setminus \mathrm{Obs}(\mathrm{S_N})|$, the anonymity notion is broken, with a similar reasoning as given for notion DSA-n-CA: The adversary can corrupt the non-observable senders in turn and run them together with $r_{c_{id}}$ through the timing oracle. If the TA-oracle has success probability less than 1, some senders might have to be run multiple times by the oracle in order to determine who is the most likely candidate. Hence, $n \leq |\mathrm{N} \setminus \mathrm{Obs}(\mathrm{S_N})| - 1$.

Note that if at least one of the challenge senders or the onion router, respectively, are observable or corrupted in [N]. Then even when $n = 1$ the network is not CDSA-n-CA, DSA-n-CA, Unlink-n-CA anonymous. The same goes for SA-n-CA when the correct sender of the circuit is observable.

As for observability, Unlink-n-OA suffers from the same problem as Unlink-n-CA: Since an adversary is not interested in the circuit id, only whether the two challenge players communicate or not. Therefore, is $n = 1$ for that notion as well.

For SA-n-OA, DSA-n-OA and CDSA-n-OA it is harder to determine if observability trivially breaks the game. Although the players in question are observable, the circuit id's are not revealed. The only way to determine whether the players communicate over the correct circuit is in principle by guessing. In those cases, n might be considered unrestricted.

To summarize, an adversary that has corrupted to much of the onion routing network has broken all anonymity notions. This stems form the fact that timing attack automatically breaks all observable/corrupted challenges with a success probability similar to its own success probability.

It is, therefore, clear that the amount of control a single adversary can have over an onion routing network has to be tightly regulated. It should also be hard to adaptively corrupt players.

7.2 Mixed Network

We will now apply the model to a mixed network we denote *general mixed network*. Mix networks were first introduced by Chaum [2]. We will here create a simplified high-latency

version of the mix network concept. The general mixed network works as follows: The network consist of a series of mixes that collect onions[2]. When enough onions are collected, the mix peels off an encryption layer off of all the onions, shuffles them around before sending the now peeled and shuffled onions to the next mix. This process continues until the onions reaches the last mix of the network. Here the messages are revealed and sent to their intended recipient. An example can be seen in Figure 7.1.

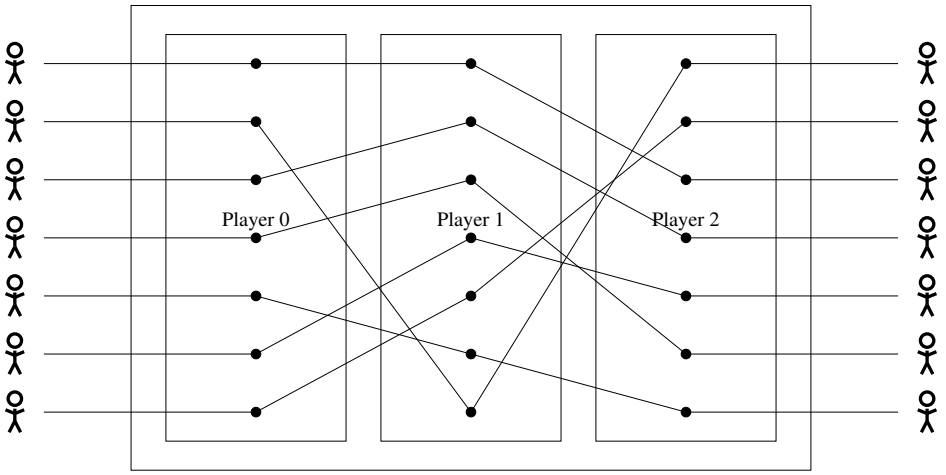


Figure 7.1: The model applied to the general mix network. The stick figures as well as the three rectangular boxes inside the network are players. The nodes are only visualizations of the onion permutations.

The mixes can be viewed as permutation functions. Meaning that an onion that enters a mix in slot n leaves from slot m. This is the same as saying each mix is a player and that every onion with incoming circuit id n has outgoing circuit id m. The network traffic is simulated in such a way that every player (mix) waits until it has a message with each incoming circuit id before sending out the outgoing messages, staring with the smallest circuit id. This means that the order onions enter a player is not the same as how they exit. In this specific network senders will always send along the same path of the mix, i.e. along the same circuit.

It is clear that due to the fact that the mixes need to wait for enough onions before any can be sent over the network, this system can be considered a high-latency network. Therefore, it is clear that the timing oracle will have success probability 0. Since, a player waits until enough packages are collected before sending them of to the next player, there is no time to do timing correlation on.

It is also clear that unlinkability is trivially broken for network players, independent of attack powers. This is due to the construction of the network, where all the network players are part of one large sequence. If an adversary has CSA power unlinkability is broken for all players, since endpoints can be related to each other as well.

Observability attack, however, even when unlimited, does not lessen the anonymity of the network. This is due to the fact that timing attacks are useless in this specific

[2]They are not actually called onions, but with respect to previous notation we will denote them as such.

network, and observing the traffic patterns to and from players reveals nothing but the size of the permutation function. Even if all the network players are observable, but not all corrupt, the system should still be DSA, SA and CDSA anonymous. Hence the general mix network is DSA-OA, SA-OA and CDSA-OA anonymous.

The general mixed network should even be DSA-n-CA, SA-n-CA and CDSA-n-CA anonymous if not all the network players get corrupted. An adversary does not know the permutation function of non-corrupted players, meaning that a senders circuit id with respect to the player should be hidden. The only exception here could be when the challenge network player is the last player in the mix sequence. If an adversary knows the receiver and sender, it can corrupt the receiver which will in turn reveal the outgoing circuit id from the last mix corresponding to the sender. However, this turns out to not be a problem since we are interested in the incoming circuit id's of the mixes, not the outgoing.

To summarize, general mix networks are DSA-atk, SA-atk and CDSA-atk anonymous, atk $\in \{$TA, n-OA, n-CA$\}$, but not unlinkable.

The results are as expected, a high-latency network such as *the general mix network* holds anonymity much better than a low-latency network like onion routing.

Chapter 8

Conclusion

The objective of this thesis has been to create anonymity notions and relate them to each other using reductions. These anonymity notions will then define a measure for how anonymous different networks are.

In order to do so, we first needed to create an ideal model for anonymity networks. This model was introduced independently of anonymity networks. The framework was later specified to fit onion routing. The traffic simulation of the model was, however, again introduced in a general way as to not be network specific.

Typical anonymity powers of low-latency networks were then introduced using attack oracles. This way time did to have to be incorporated into the model. The attack powers were then related to each other to give a hierarchy of strengths. The anonymity goals introduced were taken from Pfitzmann and Hansen [17] and altered in order to fit the onion routing model. They where then related to each other, independent of adversarial powers. The anonymity goals and attack powers were then combined into anonymity notions.

A limitation of this thesis stems from the fact that the anonymity goals and anonymity notions have been defined with onion routing in mind. Although the powers and goals should be universal and thus applicable to any anonymity network, there might prove to be some difficulties regarding this.

It was, however, shown, in Chapter 7, that the anonymity notions are still applicable to a high-latency non-onion routing based network, indicating that the anonymity notions are applicable to networks other than those based on onion routing.

In cryptography there is a well-defined model for security, indicating how different security notions relate. As far as the author is aware, no such model exist for anonymity notions. One of the greatest accomplishments of the thesis has, therefore, been the creation of anonymity notions, and thereby proposed such a model of anonymity.

Future Works

If further work on the network and anonymity models presented in this thesis is desirable, a further generalization of the anonymity goals and attack notions should be pursued.

The missing implications in Figure 6.2 should be proven or dis-proven. Receiver version of all the anonymity notions should also be related.

Finally we suggest introducing more attack powers into the model. Examples of such could be: TLS-attack; where a player is observable and the TLS-tunnels are broken, i.e. the adversary has the TLS-key, or Jurisdiction-adversary; where the network gets divided into partitions called jurisdiction and the adversary can observe any traffic traveling between these partitions [7].

Bibliography

[1] Mihir Bellare et al. “Relations among notions of security for public-key encryption schemes.” In: *Advances in Cryptology - CRYPTO ’98,* ed. by Hugo Krawczyk. Vol. 1462. LNCS. Springer-Verlag, 1998, pp. 26–45.

[2] David L. Chaum. “Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms”. In: *Commun. ACM* 24.2 (Feb. 1981), pp. 84–90. ISSN: 0001-0782. DOI: 10.1145/358549.358563. URL: http://doi.acm.org/10.1145/358549.358563.

[3] Roger Dingledine and Nick Mathewson. *Tor Path Specification*. May 2019. URL: https://gitweb.torproject.org/torspec.git/tree/path-spec.txt.

[4] Roger Dingledine, Nick Mathewson, and Paul Syverson. “Tor: The Second-Generation Onion Router”. In: *Proceedings of the 13th USENIX Security Symposium*. Aug. 2004.

[5] Roger Dingledine, Nick Mathewson, and Paul Syverson. “Tor: The Second-Generation Onion Router (2012 DRAFT)”. Nov. 2012. URL: https://people.torproject.org/~karsten/tor-design-2012-11-14.pdf.

[6] Joan Feigenbaum, Aaron Johnson, and Paul Syverson. “A Model of Onion Routing with Provable Anonymity”. In: *Dietrich S., Dhamija R. (eds) Financial Cryptography and Data Security.* Vol. 4886. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, 2007, pp. 57–71.

[7] Herman Galteland and Kristian Gjøsteen. “Adversaries monitoring Tor traffic crossing their jurisdictional border and reconstructing Tor circuits”. In: (2018).

[8] Kristian Gjøsteen. *A Brief Introduction to Symetric Cryptography*. URL: https://wiki.math.ntnu.no/_media/tma4160/symmetric-i.pdf.

[9] David M. Goldschlag, Michael G. Reed, and Paul F. Syverson. “Hiding Routing Information”. In: *Workshop on Information Hiding* (May 1996), pp. 137–150.

[10] Ilya Grigorik. *High Preformance Browser Networking*. O’Reilly Media, Inc., 2013. Chap. Transport Layer Security (TLS). URL: https://hpbn.co/transport-layer-security-tls/.

[11] Øyvind Hallsteinsen, Bjørn Klefstad, and Olav Skundberg. *Innføring i datakommunikasjon*. 2nd ed. Stifitelsen tisip og Gyldendal Norsk Forlag, 2015.

[12] Nicholas Hopper, Eugene Vasserman, and Eric Chan-Tin. “How much anonymity does network latency leak?” In: *ACM Trans. Inf. Syst. Secur.* 13 (Feb. 2010), p. 28. DOI: 10.1145/1698750.1698753.

[13] “Masterseminar in Cryptography”. 2019.

[14] S. Mauw, J. Verschuren, and E.P. de Vink. “A formalization of anonymity and onion routing”. In: *Proceedings of ESORICS 2004*. Ed. by P. Samarati et al. Sophia Antipolis: LNCS 3193, 2004, pp. 109–124.

[15] David Molnar and Michael J. Freeman. *Related Works: Anonymous Communication Systems.* 1998. URL: https://www.freehaven.net/related-comm.html.

[16] S.J. Murdoch and George Danezis. “Low-cost traffic analysis of Tor”. In: June 2005, pp. 183–195. ISBN: 0-7695-2339-0. DOI: 10.1109/SP.2005.12.

[17] Andreas Pfitzmann and Marit Hansen. *A terminology for talking about privacy by data minimization: Anonymity, Unlinkability, Undetectability, Unobservability, Pseudonymity, and Identity Management*. v0.34. Aug. 2010. URL: http://dud.inf.tu-dresden.de/literatur/Anon%5C_Terminology%5C_v0.34.pdf.

[18] Roman Pramberger. *Layer 7 - Application Layer*. May 2019. URL: https://osi-model.com/application-layer/.

[19] Chi Square. *Anonymity Networks. Don’t use one, use all of them!* 2012. URL: https://null-byte.wonderhowto.com/news/anonymity-networks-dont-use-one-use-all-them-0133881/.

[20] *The Tor Relay Guide*. May 2019. URL: https://trac.torproject.org/projects/tor/wiki/TorRelayGuide.

[21] *Tor*. Mar. 2019. URL: https://2019.www.torproject.org/.

[22] *Tor Metrics*. May 2019. URL: https://metrics.torproject.org/.