UCLouvain

École polytechnique de Louvain

# Exploring techniques for constraining outputs of network events to guarantee users' privacy

Author: **Tom Mansion**
Supervisors: **Tom Barbette, Axel Legay**
Reader: **Benoît Macq**
Academic year 2022-2023

**Abstract**

In an increasingly connected world, it is becoming more and more difficult to ensure one's privacy. Data is being collected at all times, whether it be for statistical studies, recommendation engines, or malicious purposes. Most of us recognize the need for privacy as a fundamental human right but in today's world, access to technology and information is equally important. In order to maintain our access to information through the internet in a secure and efficient way, network administrators have the need to monitor their networks for faults and security events. Such monitoring of network activity can obviously lead to breaches in the users' privacy. Previous works have shown the difficulty of anonymizing network data for its release. Thanks to the Retina framework, we can think of network privacy in a different manner by attempting to render analyses of the network data private instead of relying on faulty techniques for anonymizing packet traces. In this thesis, we discuss the applicability of different methods to protect the users of a network from privacy breaches. We compare the traditional methods of deidentification with more recent techniques, and we propose different ideas as a basis to explore in future works on the subject.

# Chapter 1

# Introduction

The privacy literature, as well as privacy regulations such as the European Union's General Data Protection Regulation and the United State's Family Educational Rights and Privacy Act often focus on the problem of rendering a dataset anonymous. The literature calls that process deidentification and it is often implemented through Quasi-Identifier-based techniques such as k-anonymity [10], l-diversity, or t-closeness. However, these techniques were designed for anonymizing data in large databases prior to their release. When dealing with data that comes from a stream, such as data coming from a network monitoring tool, these techniques are highly impractical and often ineffective. That is the reason why we chose a different approach. Instead of collecting data and trying to render it anonymous, we focused on the analyses that would be made with the data and tried to provide a way for these not to reveal information about the individuals in the dataset.

In order to do so, we imagined a few different use-cases where a network operator or a data analyst would extract some data from the network to perform some analysis.

## 1.1 Use-cases

Here we define a few different analyses that could be performed by a network operator in order to gain a better understanding of the general behavior of the users of the network. This list is certainly not exhaustive but each use case can give us a better understanding of the problems that may arise when dealing with this type of data.

**Ranking the most visited websites** This use-case is quite self-explanatory, the network operator is trying to get an understanding of the websites that are most visited by the users. Since most, if not all, of the web traffic is carried by

the TLS protocol, the operator can use TLS's *Server Name Indication* extension to identify the domain that is reached by the client. The operator first creates a list of all the domain names that are seen in the *SNI* field of Client Hello packets while simultaneously counting the different connection requests, then he sorts that list by count to obtain a ranking.

**Identifying the different IP addresses that are reached by the users of the network** In this use-case, the network operator is trying to identify patterns in the different IPs that are reached from the network. Maybe some of them are part of the same subnetwork and the operator would like to get more insight into the connections made to that specific subnetwork, maybe he's simply trying to geolocate them to understand if the network is communicating with politically tense regions of the world. The reasons behind this kind of analysis could vary but we chose this use-case to highlight some privacy issues that could arise, as we will show later in this paper.

**Detecting ongoing attacks on the network** Here, the operator is trying to detect an ongoing attack in order to either try and disrupt it or to simply document it in case of a failed attack. This type of analysis is quite more complex and depends on the specific attack but we chose to highlight it with a well-known ransomware: WannaCry. We chose this specific malware because it can be detected by network Indicators of Compromise (IoC) but the conclusions we will make later apply to all types of malware that can be detected over the network.

# Chapter 2

# Background

## 2.1 The Retina Framework

Retina is a high-bandwidth network monitoring framework that is capable of running on commodity hardware and doesn't require highly specialized equipment to set up. It was developed by Gerry Wan et al. and introduced in their 2022 paper [12]. The framework is implemented in the Rust programming language and differs from other network monitoring tools such as Snort and Zeek (formerly named Bro) [9, 8] by its efficiency and high-bandwidth capacities. Indeed, these older systems were not made to analyze network data in high speed environments and don't scale very well to today's standards.

The Retina framework allows its users to subscribe to network events and to provide callback functions to handle these events. The subscriptions are defined by a filter that provides a subset of network traffic to the callback function. This filtering mechanism is what allows Retina to process network traffic at speeds of 100Gbps on a single commodity server by discarding out-of-scope traffic as early as possible and thus reducing the computational load.

Retina's design goals are to enable complex analyses on network data by providing its users with the capability of analyzing raw packets, reassembled flows, and parsed application-layer sessions. These analyses must be feasible on high-speed networks (100+ Gbps) and the framework must be able to run on a single commodity server. The framework must be easy to use and readily deployable in a standard environment. It must be secure from malicious network data that might compromise its internal state by performing its packet analysis and flow reconstruction in a safe manner. The user-defined functions must also be memory-safe in order to avoid compromizing the security of the whole framework when running an experiment.

## 2.2 Privacy Mechanisms

### 2.2.1 Classical anonymity

#### $k$-Anonymity

This anonymity technique is one of the first to appear in the privacy literature and was introduced by Sweeney et al. [11] in their 1998 paper. The paper describes the inefficiency of de-identification techniques used by data holders to share information about a population without disclosing any individual's identity and introduces a new anonymization concept: k-anonymity. Previously, data holders such as municipalities would share or sell information about their citizens by simply removing any explicitely identifying information from their datasets such as the names of individuals or their social security number. The paper shows that this technique is not sufficient to protect the individuals because of a concept called *quasi-identifiers* - information that does not explicitely identify an individual but that represents their characteristics such as their age, gender, height or religious orientation. These quasi-identifiers, when taken together, can in turn be used to re-identify individuals in the dataset as they can form a kind of fingerprint of the individual and this process is made easier with the help of external knowledge about the individuals. K-anonymity is a technique that seeks to reduce the risk of re-identification by means of **suppression** and **generalization**. The first means that a k-anonymous database has records that are simply deleted, and the second means that some values are put into ranges rather than being discrete values. To understand the inner workings of this method, we must first understand how the authors categorize different types of data. Before applying any algorithms to a dataset, we must first split the records into different categories. The first category is the **explicit identifier**, something that directly ties a row of data to an individual such as a name or a social security number. The next category is a **sensitive attribute**. This is the record that we're trying to protect and it varies from one database to the other. For example, in a medical records database, the sensitive attribute would be the type of disease that affects a patient. All other columns of a table are called **quasi-identifiers**, or QID for short. [11] simply defines these as the attributes in a table whose release must be controlled. They are the fields that could lead to the re-identification of the people inside the database by cross-referencing them with other data sources.

Let's take an example from wikipedia that highlights this problem. Table 2.1 is an example table containing fictional patient records of a fictional hospital that performed a study on April 30th of this year. In this table, there are 8 columns and 10 rows. The name column is an explicit identifier, age, gender, height, weight, state of domicile and religion are QIDs as they do not explicitely identify the person

concerned but describe their characteristics. The disease column is a non-identifying sensitive attribute. If we simply remove the explicit identifiers from the table, it is still relatively easy for an attacker to re-identify the people concerned with the help of external information. For example, if they know that Kishor visited the hospital on April 30th and they know that he is 180cm tall and that he weighs roughly 80kg, they could deduce that Kishor has a heart-related disease. This clearly shows that simply removing explicit identifiers from the dataset is not sufficient to protect the individuals' identities.

| **Name** | **Age** | **Gender** | **Height** | **Weight** | **State of domicile** | **Religion** | **Disease** |
|---|---|---|---|---|---|---|---|
| Ramsha | 30 | Female | 165 cm | 72 kg | Tamil Nadu | Hindu | Cancer |
| Yadu | 24 | Female | 162 cm | 70 kg | Kerala | Hindu | Viral infection |
| Salima | 28 | Female | 170 cm | 68 kg | Tamil Nadu | Muslim | Tuberculosis |
| Sunny | 27 | Male | 170 cm | 75 kg | Karnataka | Parsi | No illness |
| Joan | 24 | Female | 165 cm | 71 kg | Kerala | Christian | Heart-related |
| Bahuksana | 23 | Male | 160 cm | 69 kg | Karnataka | Buddhist | Tuberculosis |
| Rambha | 19 | Male | 167 cm | 85 kg | Kerala | Hindu | Cancer |
| Kishor | 29 | Male | 180 cm | 81 kg | Karnataka | Hindu | Heart-related |
| Johnson | 17 | Male | 175 cm | 79 kg | Kerala | Christian | Heart-related |
| John | 19 | Male | 169 cm | 82 kg | Kerala | Christian | Viral infection |

Table 2.1: Patients treated in the study on April 30th [1]

To remediate this problem, the authors of [11] introduce the concept of k-anonymity. The definition is as follows:

**Definition 1** *Let $T(A_1, ..., A_n)$ be a table and $QI_T$ be the quasi-identifiers associated with it. $T$ is said to satisfy k-anonymity $iff$ for each $QI \in QI_T$ each sequence of values in $T[QI]$ appears at least with $k$ occurences in $T[QI]$.*

This simply means that each combination of QIDs that appear in the table must appear at least k times. Table 2.2 is an anonymized version of Table 2.1 with respect to the age, gender and state of domicile of the individuals. The table has 2-anonymity in this regard and helps highlight a fundamental problem with this privacy mechanism. In this table, the two different methods for achieving k-anonymity were used, namely **suppression** and **generalization**. The first was used in the name and religion columns by simply replacing the values with an asterisk. The second was used in the age column by putting the ages of the patients into ranges rather than leaving them as discrete values. As we can see, the 2-anonymity property is respected w.r.t the age, gender and state of domicile but the height and weight attributes were left untouched because in this example, we consider them as non-identifying. The problems we highlighted with the first table remain and this shows the main issue with this method: there is no way to mathematically determine if an attribute is an explicit identifier, QID, or sensitive

attribute. It may seem like the data holder simply made a wrong decision in this case as the height and weight of a person are definitely QIDs, but in more complex scenarios, the distinction is not always as easy and depends on the auxiliary knowledge the attacker might have which is hard to model.

| Name | Age | Gender | Height | Weight | State of domicile | Religion | Disease |
|---|---|---|---|---|---|---|---|
| * | 20 <Age $\leq$ 30 | Female | 165 cm | 72 kg | Tamil Nadu | * | Cancer |
| * | 20 <Age $\leq$ 30 | Female | 162 cm | 70 kg | Kerala | * | Viral infection |
| * | 20 <Age $\leq$ 30 | Female | 170 cm | 68 kg | Tamil Nadu | * | Tuberculosis |
| * | 20 <Age $\leq$ 30 | Male | 170 cm | 75 kg | Karnataka | * | No illness |
| * | 20 <Age $\leq$ 30 | Female | 165 cm | 71 kg | Kerala | * | Heart-related |
| * | 20 <Age $\leq$ 30 | Male | 160 cm | 69 kg | Karnataka | * | Tuberculosis |
| * | Age $\leq$ 20 | Male | 167 cm | 85 kg | Kerala | * | Cancer |
| * | 20 <Age $\leq$ 30 | Male | 180 cm | 81 kg | Karnataka | * | Heart-related |
| * | Age $\leq$ 20 | Male | 175 cm | 79 kg | Kerala | * | Heart-related |
| * | Age $\leq$ 20 | Male | 169 cm | 82 kg | Kerala | * | Viral infection |

Table 2.2: Patients treated in the study on April 30th (anonymized) [1]

### $\ell$-Diversity

The problem presented in the last section is definitely not the only problem with k-anonymity. Before explaining other problems, we must first define what an equivalence class means. An equivalence class is a group of rows that share the same QIDs. A k-anonymous database has equivalence classes of size at least k. We will now explain two different attacks on k-anonymity that l-diversity attempts to solve.

**Homogeneity Attack** Now, let's suppose that an attacker knows someone is inside a k-anonymous dataset that contains sensitive information, like the one shown before that includes the type of disease that afflicts each patient. If the attacker has knowledge of this person's QIDs, they can locate the equivalence class to which they belong. If all the rows in that equivalence class share the same sensitive attribute, the attacker then learns the disease that affects that person and breaks their privacy. If the equivalence class is of size 4 and there are only two different types of sensitive attributes in that class which are distributed evenly, then the attacker learns that there is a 50% chance that the person is afflicted with the first, and 50% chance that they suffer from the second. They can reduce the uncertainty with the help of background knowledge.

**Background Knowledge attack** The authors of [6] explain this attack with a simple example. Let's say an attacker is friends with a Japanese person, knows their QIDs and knows they have been to the hospital on April 30th. They can

locate the equivalence class to which they belong in the k-anonymous database and thus know that their friend's information is located in one of four records. In that equivalence class, the patients suffer either from a virus or from a heart disease. However, the attacker knows that heart-related conditions are extremely rare in Japanese people. Therefore, they can almost be certain that their friend suffer from a viral infection.

**Definitions** Introduced by Ashwin Machanavajjhala et al. in their 2006 paper [6], $\ell$-diversity is a principle that ensures that sensitive attributes are "well-represented" within an equivalence class. A table is said to be $\ell$-diverse if all of its equivalence classes are themselves $\ell$-diverse. The authors put forth two definitions of "well-represented". The first one uses the Shannon entropy and requires that it is above a certain threshold. The second one is less conservative and makes sure that the most frequent values do not appear too often in the equivalence class while the less frequent value do not appear too rarely.

**Entropy $\ell$-diversity** A table is entropy $\ell$-diverse if for every equivalence class

$$-\sum_{s \in S} p(v) log(p(v)) \geq log(\ell)$$

where $S$ is the domain of possible sensitive attributes, and

$$p(v) = \frac{n(v)}{\sum_{w \in S} n(w)}$$

where $n(v)$ is the amount of occurences of value $v$ in a sensitive attribute for an equivalence class. The authors point out that in order to have entropy $\ell$-diversity for each equivalence class, the entropy of the whole table must be at least $log(\ell)$. This definition is too restrictive in some cases, for example if some values for the sensitive attribute are very common. This is why they came up with a more relaxed definition of $\ell$-diversity.

**Recursive $(c, \ell)$-diversity** In a given equivalence class, let $r_i$ denote number of times the $i^{th}$ most frequent value appears in the equivalence class. Given a constant $c$, the equivalence class satisfies $(c, \ell)$-diversity if

$$r_1 < c(r_\ell + r_{\ell+1} + ... + r_m)$$

where $m$ is the total amount of sensitive values in the equivalence class.

### $t$-Closeness

Despite the progress made by the authors of [6], $\ell$-diversity still exhibits some problems and is vulnerable to a couple attacks. In their paper titled $t$-Closeness: Privacy Beyond $k$-Anonymity and $\ell$-Diversity, Ninghui Li et al. expose these attacks and propose a new privacy model: $t$-closeness [7].

**Skewness Attack** $\ell$-diversity doesn't take into consideration the probability distribution of the sensitive attribute. When this distribution is skewed, we could encounter a case where the distribution of the sensitive attribute is very different within an equivalence class when compared to the distribution in the whole dataset. The authors of [7] give an example where we would have one sensitive attribute that represents the results of a test that determines whether a patient is infected with a virus. This attribute could take one of two values: positive or negative. Now, suppose that 99% of the population is negative while the remaining 1% is positive. The two values are very different in their sensitivity. One would not care if an attacker knows they are negative as they are then similar to 99% of the population, while being known to be positive would represent a way bigger privacy problem. Now imagine that we have an equivalence class within that database that has 50% of records that are negative to the virus and 50% that are positive. This equivalence class would satisfy both entropy 2-diversity and recursive $(c, 2)$-diversity for any value $c$. However, this equivalence class represents a big privacy risk as, if an attacker determined that one person belonged to that equivalence class, they could know that this person has a 50% chance of being positive to the virus, compared to the 1% chance in the whole population. Let's consider another case where we have an equivalence class that has 49 records that are positive to the virus and 1 record that is negative. This equivalence class would have a higher entropy than the whole dataset and would thus satisfy the $\ell$-diversity requirement, but it would be a catastrophic failure of privacy, as the individuals contained in this equivalence class would have a 98% chance of being positive. Furthermore, the entropy of this equivalence class would be the same if 49 records were negative and 1 were positive.

**Similarity Attack** In this attack, we consider the case where different values for the sensitive attribute are semantically similar. Let us take another example to highlight this case. Table 2.3 shows a 3-diverse table where the sensitive attributes are the salary and disease columns. In this example, an attacker who knows the QIDs of a person in the first equivalence class could infer that the person is afflicted with a stomach problem. They could not know exactly which problem the person suffers from but they still gain information on their condition. The same goes for the salary, they can't know exactly which salary the person has, but they can learn that their salary is relatively low compared to others. This happens because

$\ell$-diversity only takes into consideration the diversity of sensitive attributes but not their semantical closeness.

| ZIP Code | Age | Salary | Disease |
|---|---|---|---|
| 476** | 2* | 3K | Gastric Ulcer |
| 476** | 2* | 4K | Gastritis |
| 476** | 2* | 5K | Stomach Cancer |
| 4790** | $> 40$ | 6K | Gastritis |
| 4790** | $> 40$ | 11K | Flu |
| 4790** | $> 40$ | 8K | Bronchitis |
| 476** | 3* | 7K | Bronchitis |
| 476** | 3* | 9K | Pneumonia |
| 476** | 3* | 10K | Stomach cancer |

Table 2.3: A 3-diverse table of salary/disease

The attacks we just showed demonstrate the need for another privacy metric and this is what the authors of [7] provide. They define the $t$-closeness principle as follows:

**The $t$-Closeness principle** An equivalence class is said to have $t$-closeness if the distance between the distribution of a sensitive attribute in this class and the distribution of the attribute in the whole table is no more than a threshold $t$. A table is said to have $t$-closeness if all equivalence classes have $t$-closeness.

**Earth Mover's Distance** There are many ways to calculate the distance between probability distributions but the authors of the paper chose the Earth Mover's Distance (EMD). To explain this concept, imagine we have two piles of sand that represent two probability distributions. The sand in each pile is arranged in the same way as the probability mass at every point of the distributions. The EMD is then calculated as the minimum amount of work required to move the sand from the bigger pile of sand to the smaller one in order to recreate the bigger pile. Each grain of sand moved has a cost that is proportional to the distance it has been moved over.

This metric is interesting because it is non-negative, symmetric and can handle probability distributions that are different in size, or that have outliers. The authors explain how to compute this distance over numerical and categorical attributes and show how $t$-closeness can help prevent the similarity and skewness attacks explained earlier.

### 2.2.2 Differential Privacy

Introduced by Cynthia Dwork in 2006 [2], differential privacy is privacy mechanism that differs from the ones mentioned until now because it is not applied to a whole dataset prior to its release, but rather to the processes that release aggregate statistics about the dataset.

Her motivations came from the need to provide a robust, mathematical definition of privacy that lacked in previous work. Her original paper shows that it is impossible to guarantee privacy in the presence of auxiliary information while maintaining data utility. Differential privacy aims to quantify that trade-off between privacy and utility. The general idea of this privacy mechanism is the following: consider a statistical survey that aims to record some characteristics of a population. The querying of statistical properties over that dataset might reveal some information about the participants of the survey. In order to protext these participants, the presence or absence of one of them in the dataset should not significantly affect the result of the query. In short, a data analyst should be able to extract information about the population as a whole, but not about the individuals that are a part of it. Differential privacy is a concept that applies to randomized mechanisms, this means mechanisms for querying a database that introduce some randomness in their results. This addition of randomness, or noise, is essential for protecting the individuals' privacy.

We will now give a formal definition of differential privacy and explain its implications.

#### Definition

Differential privacy is characterized by its parameter $\epsilon$ which is the parameter for the tradeoff between privacy and utility.

A randomized function $K$ gives $\epsilon$-differential privacy if for all data sets $D_1$ and $D_2$ differing on at most one element, and all $S \subseteq Range(K)$,

$$Pr[K(D_1) \in S] \leq exp(\epsilon) \times Pr[K(D_2) \in S]$$

where the probabilities are taken over the randomness of the data querying mechanism, or randomized function.

One of the most important concepts to understand with differential privacy is the one of adjacent datasets. In the definition, $D_1$ and $D_2$ are adjacent because they differ by at most one element. This is important because it ties to the concept of an individual being a part, or not, of a dataset.

In order to satisfy the definition of differential privacy, a randomized function must inject some noise into its results. This noise is drawn at random from a probability distribution like a Laplace or Gaussian distribution. In the Laplace

distribution, we set the *scale* parameter to $\frac{1}{\epsilon}$. This way, if we take two datasets that are adjacent to each other, the probability of observing the same result when they are passed through the randomized function is proportional to $e^{\epsilon}$. Let's take an example to explain this more clearly. Say we have a database of medical records. Each row of that database contains the information of one person and we want to count the amount of people that have a specific disease, a cancer for example. An $\epsilon$-differentially private counting mechanism will count the exact amount (say, 100) of records that indicate that the person has a cancer, add noise taken from $Lap(\frac{1}{\epsilon})$ and publish the result. Now let's say an attacker wants to find out whether a specific individual, Bob, has a cancer. We do not make any assumptions on the amount of knowledge this attacker has and we simply try to quantify the amount of information they can gain from looking at this noisy count. Now let's say the output of this counting mechanism was 103 after we added the noise. There was a certain probability that we would observe this value and this probability depends on the scale of the Laplace distribution. Now let's say that Bob indeed has a cancer. If we perform the noisy count once more after removing Bob from the dataset, there is also a probability that the output will be 103 but it is slightly less likely since the real count is now 99 so the noise drawn from the distribution would have to be greater. The ratio between these two probabilities is exactly $e^{\epsilon}$. The attacker would gain a slightly better understanding of Bob's condition, but this gain is bounded by the parameter of differential privacy, $\epsilon$.

# Chapter 3

# Design

## 3.1 Anonymizing the dataset vs. the outputs of analyses

Initially, we thought that in order to protect the users' privacy, we could apply classical privacy techniques to the whole dataset before letting data analysts or network operators perform their analyses. We started by identifying the fields that could be sensitive inside of packet headers in order to apply said privacy techniques. These are the fields that we deemed risky to disclose.

1. Packet timestamp - Quasi Identifier
   Disclosing exact timestamps could lead to reidentification attacks based on device clock drift. [5]

2. TLS SNI - Sensitive Attribute
   This field is obviously sensitive as it shows what website a user has connected to.

3. TLS handshake chosen ciphersuite - Quasi Identifier
   Some devices running older hardware could not be up to date with the latest advances in cryptography and therefore use older ciphersuites. If the amount of such devices running on the network is low enough, or if the adversary (data analyst) has external knowledge about a person running one of these devices, this field could be used for a reidentification attack.

Each field is either a quasi-identifier (QID) or a sensitive attribute. Once these values were attributed, we could pass them through privacy mechanisms such as K-Anonymity, L-Diversity, or T-Closeness. We did that by parsing our datasets with the *pyshark* library and creating a list that contained these fields for each TLS handshake. We used the timestamps of the Client Hello packet, the SNI

contained in said packet, and the ciphersuite that was chosen by the server in the corresponding Server Hello packet. Once passed through the privacy algorithms, this list was transformed into a CSV file that we could then use for the first use-case.

In contrast with this approach of anonymizing the dataset, we also used techniques to make the outputs of analyses private. There are mainly three techniques: a differentially-private counter algorithm, an algorithm to obfuscate part of an IP address when the data analyst wants to output a list of IP addresses, and a technique to transform discrete values into a range of values, for the case of outputting packet timestamps.

## 3.2 Implementing our solutions into the Retina framework

All of our privacy-enhancing techniques were designed for the use-case of the Retina framework. The framework, presented in [12] allows network operators and researchers to perform advanced queries about network traffic in a given network but the framework does not take into consideration the privacy of its users. As of now, it possible to create analyses within the framework that would completely breach the privacy of the users. For example, one could write an analysis that would output the IP addresses of all the devices that accessed Netflix, or even worse, a porn site from within the network it is monitoring. While this problem is not specific to Retina (it is shared by all network monitoring frameworks), we thought it might be interesting to think about the privacy of the users and implement some solutions to protect it. While implementing these solutions directly in the framework is out of scope for this project, we kept this objective in mind throughout the course of the thesis and hope that our work can one day be useful for improving the privacy of users and that our solutions might be implemented in the framework.

Without implementing it into the framework, we developed some proof of concept tools that could be used as modules within Retina. Indeed, the four tools we developed could be used to sanitize the outputs of analyses. While most of them are not functional or do not provide any privacy guarantees, we believe that with some additional work, they could prove useful to protect the privacy of the users. Figure 3.1 shows a high level interpretation of how our work could be used. Depending on the type of data that a specific analysis outputs, it could be passed through one of our tools in order to improve the outputs' privacy preserving properties. In the figure, the dataset generation comes from our own generation process which is described in Chapter 4 but in the case of the Retina framework, the data would come directly from the network that Retina is monitoring. If the analysis requests to output a list of IP addresses, we could pass that output to

our IP subnet anonymization tool in order to protect the users. If the analysis wishes to output specific packet timestamps, we would pass it through our range module in order to prevent the fingerprinting attack described in chapter 4. Lastly, our differentially private counter could be used to count the occurence of network events in order to release the counts in a privacy-preserving fashion. Although we will see in the next chapter that doing this is quite tricky and would require a well thought-out process to ensure that outputting the information does respect the principles of differential privacy. We think this module would be the most difficult to automate, as different types of data (even though they would all be numerical data) require that different precautions be put in place. We show in the next chapter that the anonymization of the whole dataset that is used as an input for the anaylses would not yield privacy-preserving results.

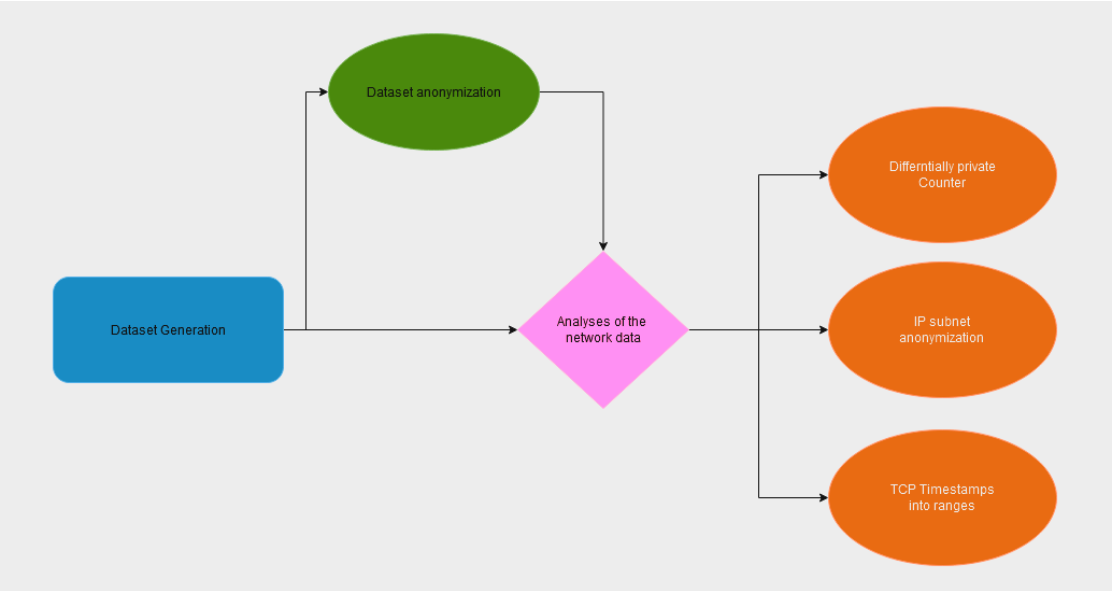


Figure 3.1: A high level visualization of how our work could be used

# Chapter 4

# Evaluation

In this section, we will compare the different privacy techniques that were mentioned in the previous section by evaluating their efficacy in protecting users' data under a few use-cases. As a reminder, the goal of this thesis is to eventually implement privacy mechanisms in the Retina framework to prevent data analysts from identifying individuals and their network behavior.

## 4.1 Dataset Generation

To perform the experiments, we needed a source of recorded network traffic. As mentioned previously, the ultimate goal is to implement the techniques shown in this section directly into the Retina framework but we decided to first implement them in the Python programming language as a proof of concept so we did all the work outside of the framework. Since there already exist many libraries for parsing packet captures (.pcap files) in Python, we decided to generate some web traffic procedurally and record it using the well-known tool *tcpdump*.

The idea was to generate web traffic coming from multiple clients in order to simulate user behavior. We started with a naïve generation process but we quickly realized that it wasn't going to be good enough to simulate a real user so we decided to go with a slightly more complex generation process.

Both generation processes start from the same point: we found a list of the top 1 million websites ranked by the alexa.com website (in 2015) that we used to make requests. That list contained quite a lot of unreachable domains because it is old so the first step was to filter it and keep only the valid domains. We wrote a python script that would attempt to make requests with the *curl* tool and discarded the domains that produced errors. Out of the list of one million domains, we extracted about 500 valid domains.

The last part of both processes was also the same. Each generation script called

the *tcpdump* system process before attempting to make requests to web domains and would save the capture file. We ran these dataset generators inside Docker containers to be able to spin up as many clients with different IP addresses as we needed. They produced one pcap file per web client and we simply had to merge them with the *mergecap* tool from the Wireshark suite in order to have a complete record of all the requests.

**Simple dataset** For this first iteration, we spun up five web clients that would make 100 requests each to random domains in the list of 500. We realized later on that this randomness did not replicate user behavior closely enough and the experiments that are described later in this section would not yield interesting results. That is why we decided to generate the dataset in a different manner.

**Complex dataset** We first decided that we needed more web clients so we bumped their number to 20 clients. Each of them would be started from a Docker container as mentioned before but we decided to wrap them up in a *docker-compose* script to make their deployment easier. Instead of making the clients connect to random websites, we tried to model real user behavior by giving each client a different subset of domains they would pick from. Each client still makes 100 requests but instead of picking them at random in the larger set of 500 domains, they would first construct their own subset of 21 domains and pick at random from that one. That subset is constructed as such: the first 15 domains are the same for every client. They are simply the first 15 domains in the list of 500. The idea behind this is that real users mostly tend to visit the same websites: social media platforms, common websites such as google.com, etc. The next five domains are a bit different. They are each picked at random in the next 20 domains in the bigger set (15th-35th). This was done to show that not all of the common websites are visited by everyone. Some people prefer reddit over twitter for example and this part of each user's subset will overlap with other users but not all of them since everyone is different. The last website in each user's subset is picked at random from the rest of the larger set (35th-500th). This domain will likely be unique for each user and we decided to construct the dataset in this way to show the limitations in the privacy techniques we show later. If only one user accesses a specific domain, the network operator should not be able to compute statistics over that connection, otherwise they would be doing it over only one user's data, which might compromise their privacy.

## 4.2 First use-case implementation: Ranking the most visited websites

This use-case might be the most important one as it is the only one where we can truly compare the different privacy mechanisms' efficiencies.

First of all, how do we know when a client connects to a specific website ? We can use the SNI extension field in TLS Client Hello packet headers as the headers themselves are not encrypted. This field contains the server name that the client wishes to connect to. So, in order to rank the most visited websites, one can count all the Client Hello packets that contain the same SNI field and sort the server names by count. Table 4.1 shows the counts for each server name taken from the raw data.

| Rank | Server Name | Count |
|---|---|---|
| 1 | www.yahoo.com | 118 |
| 2 | world.taobao.com | 103 |
| 3 | www.facebook.com | 100 |
| 4 | passport.weibo.com | 99 |
| 5 | www.google.co.in | 98 |
| 6 | www.linkedin.com | 97 |
| 7 | www.tmall.com | 95 |
| 8 | www.qq.com | 92 |
| 9 | www.youtube.com | 87 |
| 10 | twitter.com | 87 |
| 11 | www.wikipedia.org | 85 |
| 12 | www.google.com | 84 |
| 13 | www.blogger.com | 82 |
| 14 | www.sina.com.cn | 78 |
| 15 | www.google.es | 51 |
| 16 | www.bing.com | 46 |
| 17 | www.paypal.com | 45 |
| 18 | www.google.ru | 34 |
| 19 | dzen.ru | 33 |
| 20 | www.google.de | 33 |
| 21 | www.xvideos.com | 32 |
| 22 | wordpress.com | 26 |
| 23 | www.google.co.uk | 24 |
| 24 | www.apple.com | 24 |
| 25 | mail.ru | 20 |
| 26 | www.google.com.br | 20 |
| 27 | www.google.fr | 18 |

| Rank | Server Name | Count |
|---|---|---|
| 28 | www.google.it | 17 |
| 29 | www.pinterest.com | 16 |
| 30 | www.microsoft.com | 16 |
| 31 | www.google.co.jp | 15 |
| 32 | ya.ru | 12 |
| 33 | www.sohu.com | 11 |
| 34 | www.merriam-webster.com | 10 |
| 35 | www.msn.com | 9 |
| 36 | youyuan.com | 8 |
| 37 | coccoc.com | 8 |
| 38 | www.goto.com | 8 |
| 39 | www.speedtest.net | 8 |
| 40 | www.canada.ca | 8 |
| 41 | www.booking.com | 7 |
| 42 | www.irs.gov | 7 |
| 43 | www.xcar.com.cn | 6 |
| 44 | www.elmundo.es | 5 |
| 45 | www.tumblr.com | 5 |
| 46 | feedly.com | 5 |
| 47 | www.answers.com | 4 |
| 48 | www.gsmarena.com | 4 |
| 49 | www.google.ie | 4 |
| 50 | 4dsply.com | 4 |
| 51 | evernote.com | 4 |
| 52 | www.stumbleupon.com | 3 |
| 53 | www.latimes.com | 3 |
| 54 | www.huffpost.com | 3 |
| 55 | www.baidu.com | 1 |

Table 4.1: The real access counts for each server name

### 4.2.1 Attacker Model

In this use-case, we model the attacker as a malicious network operator that attempts to breach the privacy of users of the network by exposing some sensitive information about them. For example, if the network being monitored is the network of a university, they could attempt to disclose some sensitive information about the rector of the university by trying to find answers to questions such as: "has the rector of the university visited porn websites through the university network?". We chose the example of porn sites as this is obviously a sensitive matter and because we have some porn websites in our dataset but the same question could be asked for other websites like Netflix or other streaming services. The assumptions we make about the capabilities of the attacker are very different depending on the privacy mechanisms we consider. In the older models such as $k$-anonymity, $\ell$-diversity and $t$-closeness, the attacker is modeled as an entity that potentially holds external information about the individuals that are in our dataset. They then use this external information to reidentify records in an anonymized database. This definition does not make much sense in this use-case as we do not publish anonymized data but rather statistics about the data. The differential privacy definition of the attacker is better suited to this use-case. In fact, differential privacy does not make any assumptions about the attacker and their external knowledge about the population of the dataset. With differential privacy, we just have to worry about whether the attacker can gain some information with the release of the statistic. In our use-case, this would boil down to the following question: "was the rector of the university one of the users that visited that porn website?". Auxiliary knowledge about the other users should not even help the attacker to answer this question.

### 4.2.2 $K$-Anonymous data

In this section, we implement the use-case by using the data that has been anonymized wth $k$-anonymity. We passed the raw data through this algorithm with the chosen QIDs and sensitive attribute. We then counted the accesses to the different servers using the same technique as for the raw data. Table 4.2 shows the results of the count using the data that has been $k$-anonymized ($k$=2). As we can see, they are exactly the same. The problem is that the data was simply generalized with respect to the QIDs and this technique does not improve the privacy of the users in this use-case. Furthermore, we can see that the last item in the list (baidu.com) has been visited only once. This means that this value reflects the behavior of one user only and this presents a privacy risk in itself.

| Rank | Server Name | Count |
|---|---|---|
| 1 | www.yahoo.com | 118 |
| 2 | world.taobao.com | 103 |
| 3 | www.facebook.com | 100 |
| 4 | passport.weibo.com | 99 |
| 5 | www.google.co.in | 98 |
| 6 | www.linkedin.com | 97 |
| 7 | www.tmall.com | 95 |
| 8 | www.qq.com | 92 |
| 9 | www.youtube.com | 87 |
| 10 | twitter.com | 87 |
| 11 | www.wikipedia.org | 85 |
| 12 | www.google.com | 84 |
| 13 | www.blogger.com | 82 |
| 14 | www.sina.com.cn | 78 |
| 15 | www.google.es | 51 |
| 16 | www.bing.com | 46 |
| 17 | www.paypal.com | 45 |
| 18 | www.google.ru | 34 |
| 19 | dzen.ru | 33 |
| 20 | www.google.de | 33 |
| 21 | www.xvideos.com | 32 |
| 22 | wordpress.com | 26 |
| 23 | www.google.co.uk | 24 |
| 24 | www.apple.com | 24 |
| 25 | mail.ru | 20 |
| 26 | www.google.com.br | 20 |
| 27 | www.google.fr | 18 |

| Rank | Server Name | Count |
|---|---|---|
| 28 | www.google.it | 17 |
| 29 | www.pinterest.com | 16 |
| 30 | www.microsoft.com | 16 |
| 31 | www.google.co.jp | 15 |
| 32 | ya.ru | 12 |
| 33 | www.sohu.com | 11 |
| 34 | www.merriam-webster.com | 10 |
| 35 | www.msn.com | 9 |
| 36 | youyuan.com | 8 |
| 37 | coccoc.com | 8 |
| 38 | www.goto.com | 8 |
| 39 | www.speedtest.net | 8 |
| 40 | www.canada.ca | 8 |
| 41 | www.booking.com | 7 |
| 42 | www.irs.gov | 7 |
| 43 | www.xcar.com.cn | 6 |
| 44 | www.elmundo.es | 5 |
| 45 | www.tumblr.com | 5 |
| 46 | feedly.com | 5 |
| 47 | www.answers.com | 4 |
| 48 | www.gsmarena.com | 4 |
| 49 | www.google.ie | 4 |
| 50 | 4dsply.com | 4 |
| 51 | evernote.com | 4 |
| 52 | www.stumbleupon.com | 3 |
| 53 | www.latimes.com | 3 |
| 54 | www.huffpost.com | 3 |
| 55 | www.baidu.com | 1 |

Table 4.2: The access counts for each server name using the k-anonymous data

### 4.2.3 $\ell$-Diverse data

The same applies to the results computed over a 3-diverse dataset. The count of each website it identical to the real count and this technique doesn't help with the privacy of the users. The dataset is generalized even further and the utility of the data is further reduced as well.

### 4.2.4 $t$-Close data

In this case, we chose to implement 0.125-closeness in the dataset and the result is surprising. The data has been generalized so much that all of the datapoints reside in a single equivalence class. That is, all of the packets' timestamps have

been put into one big range, and all of the ciphersuites have been generalized. The quasi-identifiers for each row of data look like this:

- Timestamp: 1684576003.865055-1684576635.303198
  The timestamp is now a range that spans over the complete time range of our experiments. All of the rows in this anonymized table fall into that range.

- Ciphersuite: 0x1303,0x1302,0xc030,0x1301,0xc02f
  These five ciphersuites represent all of the possible ciphersuites that were seen in the original dataset. Each row has a ciphersuite that is in this list.

### 4.2.5 Differentially-private counter

For this variant of the use-case, we took a completely different approach. Instead of sanitizing the data before doing any computations on it, we implemented a differentially-private mechanism to perform the computations. The implementation of this mechanism comes from an implementation of Google's differential privacy library in Python. It was implemented by OpenMined and the code is open-source and available on github (https://github.com/OpenMined/PyDP/tree/dev). This repository provides several differentially-private algorithms for dealing with numerical data. These algorithms use the Laplacian mechanism to introduce noise into their calculations and protect the privacy of individuals. We will first describe how we implemented the counter mechanism and how we thought it was protecting the individuals' privacy. We realized later on that we made several mistakes in the implementation of differential privacy so we will also describe those and what we could have done better.

#### Our implementation of the counter

The idea for implementing a differentially private counter came to us after reading Dwork et al.'s 2010 paper on Differential privacy under continuous observation [3]. The paper mentions the use of differential privacy in the context of streaming algorithms and implements a counter mechanism that counts events in a stream of data in a differentially private manner. The paper goes beyond differential privacy and assumes an attacker that is capable of reading the internal state of a streaming algorithm at each step of its calculations. The authors demonstrate a counter algorithm that is resistant to this type of intrusion and thus provides an even stronger definition of privacy: pan-privacy. That is, the counter hides the true count from an eavesdropping attacker while maintaining an estimate of this count that is accurate within some bounds. Naturally, we thought this paper was extremely interesting as it fitted very well with our problem. We thought that the concept of pan-privacy was out of scope for our project but we attempted to

implement a differentially private counting algorithm nonetheless. We found this library that implemented differential privacy but we neglected a very important fact: this library was designed to perform calculations on fixed datasets, not on data that comes from a stream and the way the counter is implemented in the two settings is very different. In the first setting, if we want to count the occurences of a certain event in the dataset, in this case the number of times a specific website was accessed, the accesses are counted first and then a certain amount of noise is added to them. While in the second setting, as the counting algorithm is continuously updating its estimate of the true count, the noise is added at each iteration, every time the event occurs. This makes a big difference in the interpretation of differential privacy. The next paragraph explains how we believed the counting algorithm worked and how it protected the users of the network by adding noise at each step of the calculation. The section that follows it describes other problems with our approach.

**The flawed approach** At each step, the private counter adds some noise into the count. That noise is drawn from the Laplace distribution and since it is centered around 0, some iterations will add negative noise while some others will add positive noise. This process balances itself out in the end and that is why the values we end up with are close to the original counts. The more we count, the more this happens, and the websites that had a high amount of visits are less likely to suffer from the distortion brought by the addition of noise. This noise is drawn at random from the Laplace distribution so different iterations of the same count over the same data yield different results. Table 4.3 shows the results of the count over the same dataset with the same value for the $\epsilon$ parameter. We only show the 25 most visited websites for visibility reasons. The value for the parameter is chosen arbitrarily to demonstrate the randomness associated with the differentially private counter mechanism. Even though there is this added randomness, we can notice that the ranks do not differ too greatly from the true ranks, nor between the two iterations of the process. The process that protects the users' privacy is the addition of the random noise to the count every time an additional item is accounted for. This addition of noise give plausible deniability to the users as there is a high probability (depending on the value of $\epsilon$ and thus the amount of noise $Lap(\frac{1}{\epsilon})$ added to the count) that the count would have been the same, whether or not their input was taken into account when computing this count.

| Rank | Server Name | Count |
|---|---|---|
| 1 | www.yahoo.com | 116 |
| 2 | world.taobao.com | 103 |
| 3 | passport.weibo.com | 103 |
| 4 | www.google.co.in | 99 |
| 5 | www.linkedin.com | 97 |
| 6 | www.tmall.com | 94 |
| 7 | www.facebook.com | 92 |
| 8 | www.qq.com | 90 |
| 9 | twitter.com | 88 |
| 10 | www.google.com | 87 |
| 11 | www.youtube.com | 85 |
| 12 | www.wikipedia.org | 85 |
| 13 | www.blogger.com | 82 |
| 14 | www.sina.com.cn | 78 |
| 15 | www.google.es | 50 |
| 16 | www.paypal.com | 46 |
| 17 | www.bing.com | 40 |
| 18 | www.google.ru | 36 |
| 19 | dzen.ru | 35 |
| 20 | www.google.de | 33 |
| 21 | www.xvideos.com | 31 |
| 22 | wordpress.com | 26 |
| 23 | www.google.co.uk | 25 |
| 24 | www.google.com.br | 23 |
| 25 | www.apple.com | 20 |

| Rank | Server Name | Count |
|---|---|---|
| 1 | www.yahoo.com | 112 |
| 2 | world.taobao.com | 104 |
| 3 | passport.weibo.com | 101 |
| 4 | www.facebook.com | 100 |
| 5 | www.linkedin.com | 99 |
| 6 | www.tmall.com | 96 |
| 7 | www.qq.com | 94 |
| 8 | www.google.co.in | 92 |
| 9 | www.youtube.com | 89 |
| 10 | twitter.com | 87 |
| 11 | www.google.com | 85 |
| 12 | www.wikipedia.org | 85 |
| 13 | www.blogger.com | 81 |
| 14 | www.sina.com.cn | 77 |
| 15 | www.google.es | 51 |
| 16 | www.bing.com | 49 |
| 17 | www.paypal.com | 43 |
| 18 | dzen.ru | 36 |
| 19 | www.google.ru | 34 |
| 20 | www.google.de | 33 |
| 21 | www.xvideos.com | 32 |
| 22 | www.google.co.uk | 27 |
| 23 | www.apple.com | 27 |
| 24 | wordpress.com | 25 |
| 25 | www.google.fr | 21 |

Table 4.3: Two iterations of access counts computed with differentially private counter ($\epsilon$=0.5)

Table 4.4 shows two iterations of the counting algorithm with different values of $\epsilon$. The first is the result with an $\epsilon$ set to 0.2, and the second with $\epsilon$ set to 0.7. As we can see, a lower value for $\epsilon$ yields results that are less accurate but that better protect the users' privacy.

| Rank | Server Name | Count |
|---|---|---|
| 1 | www.yahoo.com | 116 |
| 2 | world.taobao.com | 104 |
| 3 | www.facebook.com | 102 |
| 4 | passport.weibo.com | 99 |
| 5 | www.linkedin.com | 98 |
| 6 | www.google.co.in | 97 |
| 7 | www.tmall.com | 96 |
| 8 | www.qq.com | 94 |
| 9 | www.youtube.com | 88 |
| 10 | twitter.com | 86 |
| 11 | www.google.com | 85 |
| 12 | www.wikipedia.org | 85 |
| 13 | www.blogger.com | 84 |
| 14 | www.sina.com.cn | 81 |
| 15 | www.google.es | 52 |
| 16 | www.bing.com | 47 |
| 17 | www.paypal.com | 45 |
| 18 | www.google.ru | 34 |
| 19 | www.google.de | 33 |
| 20 | www.xvideos.com | 33 |
| 21 | dzen.ru | 32 |
| 22 | www.google.co.uk | 24 |
| 23 | www.google.com.br | 24 |
| 24 | www.apple.com | 24 |
| 25 | wordpress.com | 22 |

| Rank | Server Name | Count |
|---|---|---|
| 1 | www.yahoo.com | 127 |
| 2 | world.taobao.com | 108 |
| 3 | www.facebook.com | 98 |
| 4 | passport.weibo.com | 98 |
| 5 | www.linkedin.com | 95 |
| 6 | www.tmall.com | 93 |
| 7 | www.qq.com | 91 |
| 8 | twitter.com | 89 |
| 9 | www.google.co.in | 89 |
| 10 | www.blogger.com | 87 |
| 11 | www.sina.com.cn | 86 |
| 12 | www.youtube.com | 85 |
| 13 | www.google.com | 84 |
| 14 | www.wikipedia.org | 82 |
| 15 | www.bing.com | 44 |
| 16 | www.google.de | 43 |
| 17 | dzen.ru | 42 |
| 18 | www.google.es | 42 |
| 19 | www.paypal.com | 42 |
| 20 | www.google.ru | 37 |
| 21 | www.xvideos.com | 32 |
| 22 | www.apple.com | 31 |
| 23 | www.google.com.br | 26 |
| 24 | wordpress.com | 25 |
| 25 | www.google.co.uk | 24 |

Table 4.4: Access counts computed with the differentially private counter ($\epsilon$=0.7 for the table on the left and $\epsilon$=0.2 for the table on the right)

### Mistakes we made in the usage of differential privacy

**Event-level privacy vs. user-level privacy** As we mentioned previously, several mistakes were made when implementing this differentially private counter. The first (and biggest) mistake was the confusion made between the streaming setting and the one-shot calculation setting. This resulted in a count that did not protect the users' privacy in the way we expected. Furthermore, even if the counting algorithm worked as we thought it did, it would only protect the privacy of the events themselves and not the users. Indeed, differential privacy is usually applied to databases where one row represents the contribution of one user. In

our case, each row represented a single access to a website, regardless of the user. The user's contribution to a website's access count was split across multiple rows and this completely breaks any reasoning we can make about the privacy of the users themselves. In [3], this is referred to as event-level privacy, whereas what we wanted to achieve is user-level privacy.

**Bounding the amount of contributions made by each user** We did not take into consideration the fact that a user could access a website multiple times. If that happens, let's say if one user accesses the same website (facebook.com, for example) five times where the other users only access it once, it means that this user has a bigger impact than the others in the count. In turn, this means that if we remove this user from the dataset, the probability to see the same final value for the access count of facebook.com is much lower if we use the same amount of noise in our calculations. This would yield $5\epsilon$ differential privacy, a much lower privacy guarantee than the original $\epsilon$ differential privacy. To counter this phenomenon, we must first place a bound on the amount of contributions a single user can make. Once this bound has been decided, we must then increase the amount of noise that we add to our count. Instead of adding $Lap(\frac{1}{\epsilon})$ to the count, we must add $Lap(\frac{bound}{\epsilon})$.

**Websites that are accessed by only one user should be removed** Remember Table 4.1 that shows the real access counts for each website. The last element in that table (baidu.com) was only accessed once, by a single user of course. If we remove that user from the dataset, the resulting ranking of the websites will simply not contain that website, which completely breaks differential privacy as the probability of seeing a certain count for that website, regardless of its value, will be 0. Therefore, before applying our counting mechanism, we must first remove any website that was accessed only once.

## 4.3 Second use-case implementation

In this use-case, the network operator is trying to get a list of IP addresses that are reached by the users of the network. Initially, this use-case was chosen to demonstrate the risks of outputting IP addresses from within the network, as the output of another analysis. If a network operator wanted to see which IP addresses visited a specific website for example. Obviously, this would constitute a privacy risk for the individual that's concerned. We changed the use-case to external IP addresses simply because our dataset generation process involved multiple docker containers as our clients and they all resided within the same subnetwork.

Our intuition was that instead of releasing the full IP addresses of the concerned individuals, we would only release part of it, so we would release the subnetwork in which that address resides. Of course, if that subnet only has a couple of computers inside of it, the privacy of both users could be compromised so we needed to set a threshold of users starting from which we could start releasing that information. We devised a system to keep track of the amount of different IP addresses inside each subnetwork such that, if we want to release a specific IP address, the system checks if there are enough participants inside the subnetwork before releasing it.

This works on multiple layers of the IP address. As you may know, an IPv4 address is comprized of four bytes and each byte denotes a different layer of the network architecture. Let's take an example: 179.60.195.36 is the IP address of one of facebook.com's servers. This server is inside the 179.60.195.* subnetwork, which is inside the 179.60.*.* subnetwork, and so on.

Our system keeps track of the amount of different addresses it has seen inside each subnetwork layer. It works by building a tree where each node represents a subnetwork. Figure 4.1 shows a representation of this tree. The root node does not represent any network but can hold the total count for all the addresses we've seen in the network we apply this system on. The first layer of nodes hold the value for the first layer of subnetworks, the second one holds the second layer and so on. Each node holds the count of how many different IP addresses have been seen inside that subnetwork. We do not store the last byte of each IP address as that is always an identifier. When we request a specific IP to be output, the system checks whether it has seen the subnetworks to which this address belongs. If the subnet holds a count that is superior to the threshold defined by the user, that subnetwork is allowed to be output. Then, the next subnetworks are checked in the same way until one of them doesn't hold enough addresses. In that case, the remaining part of the IP address is obfuscated and replaced with asterisks. Let's take an example: we want to output the address 179.44.12.5 and the threshold for our network is set at 5. We can output the first and second subnetworks in the tree because they hold counts that are superior to 5, but we have to stop there, as the 179.44.12.* subnetwork only holds a count of 4. The output would then be 179.44.*.*.

There is a literature for prefix-preserving IP anonymization techniques and the most known implementation of this is the one of CryptoPAn. Designed by Jinliang Fan et al. in 2004 [4], CryptoPAn is a cryptographic approach to the problem of anonymizing IP addresses while conserving their prefix. However, we chose to implement our own scheme because the goal of CryptoPAn is slighly different from ours. The tool was developed with the aim of releasing entire packet traces while conserving some properties in the IP address prefixes while our own tool rather aims at obfuscating part of an IP address prior to releasing it in the context of a network analysis. Our goal was to ensure that a released subnetwork would be

shared by enough users to hide the users in the mass. The parameter of our tool simply denotes the amount of users that should be part of a subnetwork before releasing said subnetwork. The optimal value for this parameter might vary from one network to the other, and even within the same network at different times of the day. Optimizing this parameter to find a good trade-off between privacy and utility is left for future work.

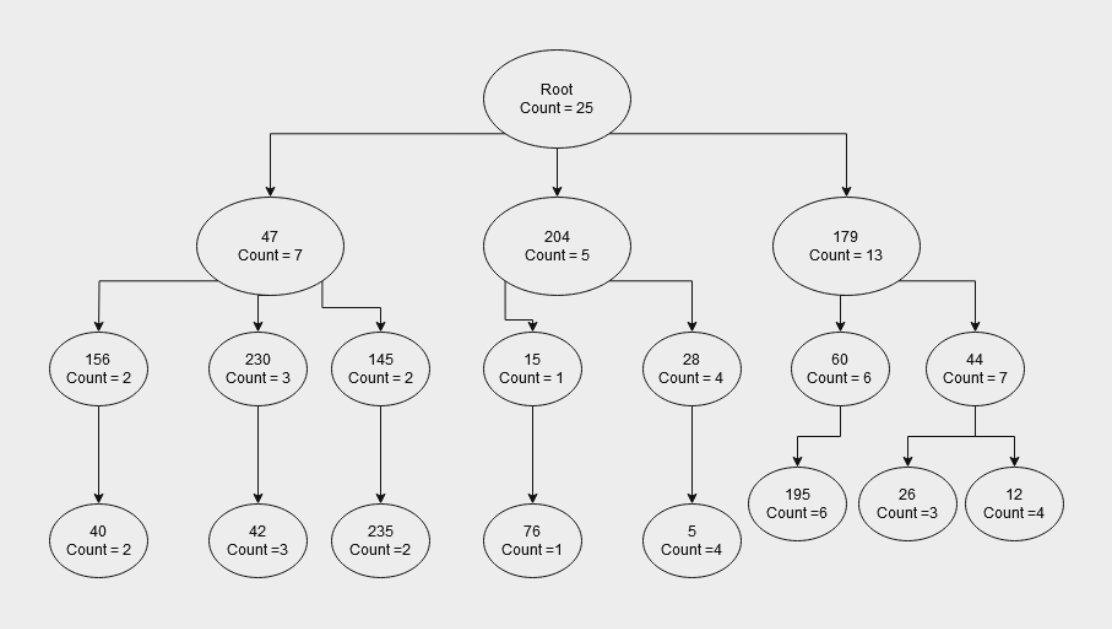


Figure 4.1: The IP tree that is built by using the network

## 4.4 Third use-case: Detecting malware on the network

For this use-case, the network operator aims to detect active malware on the network. There are multiple purposes to this task: they could be developing a signature for a specific strain of malware in order to block it from the network completely, or they could be trying to detect attempts to infect computers within the network. Another possibility is that they get prompted by a government agency to perform this analysis for them to collect statistics about a strain of malware. All these are valid reasons but we need to keep the privacy of the users of the network in mind. A person might not want to be identified as the victim of a malware because they clicked on a shady link as they could be shamed for it. So, the outcome of any analysis must be some statistics or general information about

the malware that could not be used to identify the users targeted by the attack.

We have developed three different tools to protect users' privacy and we can apply them to this problem. The first one is the differentially private counter algorithm that injects noise into the counts it makes, the second one is the IP address obfuscation technique, and the last, which we haven't covered yet, is a tool that converts a discrete numerical value into a range of values. This last tool was developed in order to prevent remote device fingerprinting attacks through packet timestamps.

This problem is covered in the 2005 paper by Kohno et al. [5] which puts forth a technique to fingerprint a device through TCP packet timestamps. Indeed, the authors of the paper show that they were able to construct signatures for specific devices by analyzing timing information from packet traces. They showed that an attacker could construct fingerprints for physical devices by hosting a website and analyzing timing-related information about the clients of the website. They showed that seemingly innocuous information such as the delay between packets, or the patterns of packet bursts could reveal some information about a device's clock drift, which in turn could be used to fingerprint the physical device. Note that we talk here about *physical device* fingerprinting, as opposed to *operating system* fingerprinting, which is another idea. Once the fingerprint has been constructed by the attacker, they could use it to reidentify users inside of an anonymized packet trace. Of course, this thesis does not cover the issue of releasing an anonymized packet trace but we feel that some analyses initiated by network operators or researchers might attempt to output TCP packet timestamps as part of their analysis, which could reveal to be a problem if the analysis itself reveals something sensitive. For example in this use-case, an operator might want to perform an analysis that outputs the timestamps (among other things) of packets that were linked to a cyber-attack. If the amount of packets linked to that attack is high enough, the attacker (in this case, the attacker is the network operator or researcher) could use these timestamps to calculate the victim's (the victim of the cyber-attack) clock drift and thus reidentify them, which we have seen could lead to a breach of privacy.

To conclude this section, the privacy issues linked with analyzing or detecting malware on a network are subtle, and strongly depend on the data that the network operator requests to be output. We have developed some tools to provide better privacy to the users of a network that is being monitored and on which analyses can be performed. The most obvious one to use in this case is the IP subnetwork obfuscation tool that was explained in the previous section. If a researcher wanted to output the IP addresses that were targeted by a cyber-attack while preserving the users' privacy, they could use our IP obfuscation tool to make sure they wouldn't reveal individual users' IP addresses while still capturing some relevant information

about the subnetwork in which they reside. Of course, as long as that subnetwork is populated by enough users that the victim of the cyber-attack is be protected by the mass. The differential privacy tool that we mention in the second section of this chapter (if implemented properly) is quite versatile and is very useful for releasing good estimates of statistics. In this use-case, it could be used to count the amount of targets of a cyber-attack (provided that the operators are able to detect that cyber-attack) while maintaining the targets' privacy as differential privacy ensures that the statistic remains roughly the same whether any individual joins or opts out of the dataset.

# Chapter 5

# Conclusion

This thesis was an attempt at solving some privacy issues linked with the monitoring and analysis of network data. We began by analyzing the different types of data that could be extracted from TLS packets and the privacy risks associated with them. We attempted to apply classical anonymization algorithms to this data in order to see if we could reduce those risks. We explored the viability of those techniques and confronted them to real use-cases. We concluded that these techniques were not suited to reduce the risks associated with these use-cases and attempted to solve these problems in different ways. We then explored another type of privacy enhancing technologies that were more recent and were confronted with the real-world difficulties of implementing them properly. Indeed, differential privacy is a complicated topic that requires a deep understanding of the literature in order to make it fit to our problem. Nonetheless, we attempted to implement it in a way that was ultimately unsuccessful but that allowed us to understand it better and to highlight some issues that could arise when dealing with this type of data. Future work on the topic could benefit from our learnings and not fall into the same traps we encountered. We also came up with custom solutions to the problems linked with releasing IP addresses and packet timestamps. Our solution to the first problem allows network operators and researchers to gain valuable information about the subnetworks in which an event occurs without breaching the privacy of the individuals that reside in the network. This solution does not give any mathematical guarantees regarding the privacy of the users but is an attempt at balancing privacy and utility. Our solution to the second problem, while rather trivial, could also allow network operators and researchers to gain valuable information, this time regarding the timing of certain events, while protecting the users from being identified through a seemingly innocuous piece of data: the timestamps in TCP packets.

In conclusion, we learned a lot about the world of privacy and anonymity and this work allowed us to ask ourselves some important questions that might arise

in our field of work in the future. In an increasingly connected world, where our data is collected every second, maintaining our right to privacy is becoming more and more important and this work shows how fragile the concept of privacy is. It is paramount that we keep researching this field and devising new mechanisms to protect ourselves from malicious actors.

# Bibliography


[1] The wikipedia k-anonymity page. `https://en.wikipedia.org/wiki/K-anonymity`

[2] Dwork, C.: Differential privacy. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) Automata, Languages and Programming. pp. 1–12. Springer Berlin Heidelberg, Berlin, Heidelberg (2006)

[3] Dwork, C., Naor, M., Pitassi, T., Rothblum, G.N.: Differential privacy under continual observation. In: Proceedings of the Forty-Second ACM Symposium on Theory of Computing. p. 715–724. STOC '10, Association for Computing Machinery, New York, NY, USA (2010)

[4] Fan, J., Xu, J., Ammar, M.H., Moon, S.B.: Prefix-preserving ip address anonymization: measurement-based security evaluation and a new cryptography-based scheme. Computer Networks 46(2), 253–272 (2004)

[5] Kohno, T., Broido, A., Claffy, K.: Remote physical device fingerprinting. IEEE Transactions on Dependable and Secure Computing 2(2), 93–108 (2005)

[6] Machanavajjhala, A., Gehrke, J., Kifer, D., Venkitasubramaniam, M.: L-diversity: privacy beyond k-anonymity. In: 22nd International Conference on Data Engineering (ICDE'06). pp. 24–24 (2006)

[7] Ninghui Li, Tiancheng Li, S.V.: t-closeness: Privacy beyond k-anonymity and

[8] Paxson, V.: Bro: A system for detecting network intruders in real-time. Comput. Netw. 31(23–24), 2435–2463 (dec 1999)

[9] Roesch, M.: Snort - lightweight intrusion detection for networks. In: Proceedings of the 13th USENIX Conference on System Administration. p. 229–238. LISA '99, USENIX Association, USA (1999)

[10] Samarati, P., Sweeney, L.: Generalizing data to provide anonymity when disclosing information. Tech. rep. (mar 1998)

[11] Samarati, P., Sweeney, L.: Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression (1998)

[12] Wan, G., Gong, F., Barbette, T., Durumeric, Z.: Retina: Analyzing 100gbe traffic on commodity hardware. In: Proceedings of the ACM SIGCOMM 2022 Conference. p. 530–544. SIGCOMM '22, Association for Computing Machinery, New York, NY, USA (2022)

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