Summer 7-2021

Efficient Yet Robust Privacy for Video Streaming

Luke Cranfill
Kennesaw State University

Junggab Son

Efficient Yet Robust Privacy for Video Streaming

A Thesis Presented to

The Faculty of the Computer Science Department

by

Luke Cranfill

In Partial Fulfillment

of Requirements for the Degree

Master of Science, Computer Science

Kennesaw State University

July 2021

Efficient Yet Robust Privacy for Video Streaming

Approved:

DocuSigned by:
Junggab Son
A197BD0D60714A7...
July 31, 2021

Dr. Junggab Son – Advisor

DocuSigned by:
Coskun Cetinkaya
028A87E4965A4EE...
August 2, 2021

Dr. Coskun Cetinkaya – Department Chair

DocuSigned by:
Sumanth Yenduri
B04458D098CE4E8...
August 2, 2021

Dr. Sumanth Yenduri – Interim Dean

In presenting this thesis as a partial fulfillment of the requirements for an advanced degree from Kennesaw State University, I agree that the university library shall make it available for inspection and circulation in accordance with its regulations governing materials of this type. I agree that permission to copy from, or to publish, this thesis may be granted by the professor under whose direction it was written, or, in his absence, by the dean of the appropriate school when such copying or publication is solely for scholarly purposes and does not involve potential financial gain. It is understood that any copying from or publication of, this thesis which involves potential financial gain will not be allowed without written permission.

_____

Your Name

Notice To Borrowers

Unpublished theses deposited in the Library of Kennesaw State University must be used only in accordance with the stipulations prescribed by the author in the preceding statement.

The author of this thesis is:

Luke Cranfill

The director of this thesis is:

Dr. Junggab Son

Users of this thesis not regularly enrolled as students at Kennesaw State University are required to attest acceptance of the preceding stipulations by signing below. Libraries borrowing this thesis for the use of their patrons are required to see that each user records here the information requested.

Efficient Yet Robust Privacy for Video Streaming

An Abstract of

A Thesis Presented to

The Faculty of the Computer Science Department

by

Luke Cranfill

Bachelor of Biology, University of Georgia, 2018

In Partial Fulfillment

of Requirements for the Degree

Master of Science, Computer Science

Kennesaw State University

July 2021

Abstract

MPEG-DASH is a video streaming standard that outlines protocols for sending audio and video content from a server to a client over HTTP. The standard has been widely utilized by the video streaming industry. However, it creates an opportunity for an adversary to invade users' privacy. While a user is watching a video, information is leaked in the form of meta-data, the size and time that the server sent data to the user. This information is not protected by encryption and can be used to create a fingerprint for a video. Once the fingerprint is created, the adversary can use this to identify whether a target user is watching the corresponding video. Successful attack schemes have been proposed based on this leakage of user data using both Machine Learning (ML) and algorithmic approaches. Only one defense strategy has been proposed to deal with this problem: using differential privacy that adds a sufficient amount of noise in order to muddle the attacks. However, this strategy still suffers from the trade-off between the privacy level and efficiency for both the server and the client. To break through the problem, this paper proposes two schemes. A server-side defense and a client-side defense against the attacks with rigorous privacy and performance constraints, creating a totally private, scalable solution that outperforms the extant schemes. Our two proposed schemes, No Data are Alone (NDA) and a proposed scheme that uses only a single cluster (Single Cluster Solution), are developed based on K-Means clustering and are highly efficient. The experimental results show that our schemes are more than two times as efficient, in terms of excess downloaded video (represented as waste), than the most efficient differential privacy-based scheme. Additionally, no classifier

can achieve an accuracy above 7.07% against videos obfuscated with our scheme *NDA* and 2.5% against our Single Cluster Solution.

Efficient Yet Robust Privacy for Video Streaming

A Thesis Presented to

The Faculty of the Computer Science Department

by

Luke Cranfill

In Partial Fulfillment

of Requirements for the Degree

Master of Science, Computer Science

Advisor: Dr. Junggab Son

Kennesaw State University

July 2021

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Server to client video streaming is commonly encrypted and is characterized by a series of requests from client to server, and subsequent fulfillment of these requests from server to client. Popular online streaming services, such as YouTube, Netflix, etc., all share the industry standard MPEG-DASH, a protocol for server to client video streaming over HTTP. Chosen ubiquitously in the industry for its high performance, in spite of the widespread use of cryptography today, the standard has a weakness: it can be exploited by a side channel attack, allowing for an adversary to compromise user privacy by determining whether or not a user is streaming any video chosen by the adversary. With YouTube being used for both recreation and as an educational hub, there are many things a user might not want to be exposed. It is possible for an adversary to steal sensitive information about a user's health, personal relationships, possessions, or future actions, including, for example, how to make a house appear occupied while on vacation.

One of the components of MPEG-DASH that allows it to become an effective attack surface is the reliance on variable bit-rate encoding (VBR). Bit-rate is the measure of bits per second being sent across a system, in the case of video streaming it is the amount of bits needed to encode one second of video that is sent from server to client. This number of bits can be fixed, Constant Bit-rate Encoding (CBR), or vary depending on the content to be

sent. VBR is a double-edged sword. It allows for efficient use of storage and high quality streaming, but also allows a unique fingerprint to be made for a video. VBR only sends as many bits as needed to render each segment of video, making it far less wasteful than CBR and is the reason it is widely used instead of CBR. In a video encoded by VBR, a high action scene will require more bits and have a relatively higher bit-rate, and a lower action scene a lower bit-rate. MPEG-DASH breaks videos into time segments of approximately the same length (Sodagar, 2011), and a client will only request a new video segment when its buffer falls below the threshold. Thus, a client creates uniquely sized bursts of traffic over time, which can be used as a fingerprint for a video. Researchers have created various attack models based on this information (Schuster, Shmatikov, & Tromer, 2017; Gu, Wang, Yu, & Shen, 2019; Reed & Klimkowski, 2016; Dubin, Dvir, Pele, & Hadar, 2017).

The most effective of these attacks is by Schuster *et al.* (Schuster et al., 2017), its effectiveness is due to the fact that it makes no closed world assumptions, has high accuracy, and because it can be executed by JavaScript code (e.g., in the form of a malicious web browser advertisement). This attack relies on a Convolutional Neural Network (CNN) that is trained on the meta-data of the target video to be identified, and other video bit-rate measurements are used for negative examples. The adversary measures video stream bursts by saturating the network connection between the client and the server and then estimating the change in congestion; a form of timing side channel attack used against schedulers (Kadloor, Kiyavash, & Venkitasubramaniam, 2016). This saturation allows the adversary to learn the victims traffic pattern, and consequently, the video burst pattern. This attack was used to great effect, and the YouTube video classifier from the paper had 98.8% recall and 0 false positives (Schuster et al., 2017).

The defense to these attacks is straightforward in principle, but its implementation requires careful consideration because of the potential computational overhead. To stop the bit-rate streaming pattern from being able to be identified by an adversary, the streaming pattern must be obfuscated. To the best of our knowledge, the only defense algorithm was

proposed by Zhang *et al.* (Zhang, Hamm, Reiter, & Zhang, 2019). This paper focused on defending against the CNN attack model mentioned above (Schuster et al., 2017). The work done in this paper uses differential privacy, specifically $d^* - privacy$ (Xiao, Reiter, & Zhang, 2015), which is adjusted for time series data, and the Fourier Perturbation Algorithm (FPA_k) which was proposed by Rastogi *et al.* (Rastogi & Nath, 2010). The goal of the defense is to create an obfuscated bit-rate pattern with differential privacy and then use a proxy in the form of a browser extension to send segment requests based on this differentially private pattern. These methods were able to successfully reduce the accuracy of the CNN model below 50%, but incurred waste in the form of extra downloaded material, or ran a deficit by not downloading enough material. Differential privacy always trades a lack of utility in exchange for privacy, in this case, the waste incurred by this solution is a hindrance when watching video streams, especially on already computationally weak mobile devices.

In light of the computational constraints of many users, and seeking to find a bit-rate request pattern that was not random, but efficient, we pursued K-Means clustering. The centroid of a cluster in K-Means clustering would provide us with an average of all videos in that cluster, and so this pattern would be representative of many videos, so it can be used to obfuscate efficiently by replacing a video with its centroid pattern. Sometimes a cluster may only have one data point (video), making the cluster's centroid equal to the video in the cluster. Obfuscation with this centroid would provide no privacy. Because of this, K-Means cannot be used without augmentation.

Our proposed server-side defense scheme and our client-side defense schemes are an augmented version of K-Means, and they cluster videos based on bit-rate over time. We then use the cluster centroids as the new pattern for video requests to be sent, creating an efficient request pattern. Our privacy is shown through experimentation and through a formal privacy definition. We recreate the CNN video classifier (Schuster et al., 2017) to show our scheme's effectiveness. In addition to the experimentation done to show our scheme's

privacy, we give a formal privacy definition and a formal distance privacy definition that is based in the L_1 norm, in order to give a broader view of privacy beyond just one attack. When compared to differential privacy, our methods significantly reduces computational waste while providing higher privacy.

The contributions of our paper are summarized as follows:

- An effective and novel server-side defense scheme that generates an optimal request pattern called No Data are Alone (*NDA*) is proposed
- An effective and novel client-side defense scheme that generates an optimal request pattern is proposed
- The time complexity of our server-side scheme is compared with the differential privacy schemes, which hasn't been shown previously for the differentially private schemes
- Multiple attack CNNs are created and trained on un-obfuscated data and noised data and detailed explanations of how they are trained are given
- These CNNs are then used for a thorough evaluation of privacy provided by our scheme compared to differentially private schemes
- The privacy of both *NDA* and the differentially private schemes is evaluated using a distance privacy definition based on the L_1 norm, the results show that our scheme outperforms differential privacy
- A formal privacy definition is defined for our schemes to give a scope of privacy beyond experimentation.

The remainder of this paper is organized as follows: Section 2 gives necessary background knowledge, then Section 3 lays out the problem description. Section 4 introduces our proposed scheme *NDA* in detail, and Section 5 details implementation and experimental

methods. Section 6 shows the results of our evaluations on *NDA*. Section 7 introduces our Single Cluster solution and details the experimental results from this scheme. Section 8 introduces some notable related research results. Finally, we conclude this paper in Section 9.

Chapter 2

Background

2.1 MPEG-DASH

MPEG-DASH is a ubiquitous standard for video streaming, employed by companies like Netflix and YouTube. MPEG-DASH begins a streaming session by sending a Media Presentation Description (MPD) to the client. The MPD is an XML file that outlines the video segments available for each quality level, along with other characteristics needed for streaming. The DASH client then parses this file and determines the appropriate quality, segments to request, and other information. Then it begins streaming using HTTP GET requests (Sodagar, 2011).

MPEG-DASH uses VBR, an often used means of encoding video streams because of its efficiency. VBR encodes only as much of a video file as is necessary. Meaning that scenes in a video have a comparatively higher or lower bit-rate depending on what takes place in the stream at that point in time. DASH mandates that the video is streamed in segments, each being requested when a user falls below their buffer threshold. The segment sizes (bits) are based on video display time. Video display time can be variable sizes or held constant (Sodagar, 2011), but is most often held constant. Because of variable bit-rate encoding, each segment contains a different amount of bytes.

In addition to using VBR and standardized segment sizes with MPEG-DASH, video streaming is bursty (Rao et al., 2011) because there are periods when new segments are being requested which cause a spike in bits being sent from server to client, then there are break periods where no bits are requested. This combination of variable bit-rate segments, size standardized segments, and bursty segment request patterns led researchers to develop a successful traffic analysis attack that use this data as a fingerprint. A visualization of a video's un-obfuscated bit-rate over time can be seen in Figure 4.1.

2.2 K-Means Overview

K-Means is a popular type of unsupervised learning. The term K-Means comes from (MacQueen et al., 1967), thought the original algorithm is credited to Steinhaus Hugo (Steinhaus, 1956). The algorithm that is commonly used for the implementation of K-Means today, and the algorithm used by Sklearn [1] which is the Pyhton package used for experimentation in this paper, was developed by Stuart Lloyd (Lloyd, 1982). It involves the formation of clusters in data. Specifically, it involves the formation of any number of clusters, determined by the number k. This value is set by the user of the algorithm. With a value of $k = 2$, two clusters will be created.

At the center of each data cluster is a cluster centroid. This cluster centroid is the mean of all of the values that belongs to that cluster, which is where the algorithm derives its name. It will posses k number of means (cluster centroids).

The first step in this algorithm is the initialization of cluster centroids. Initializing a centroid means giving it a value within the dimensions of the data-set. These cluster centroids can be randomly initialized or use an algorithm such as kmeans++. Once the cluster centroids are initialized, the Euclidean distance is calculated from each cluster centroid to each datapoint. The Euclidean distance between two data p and q in n dimensions is

[1] https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

calculated by:

$$d(p,q) = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2} \tag{2.1}$$

Once the Euclidean distance is calculated, each data point is assigned to a cluster. This assignment is based on the Euclidean distance to each cluster centroid. Whichever centroid is closest to a data point is the centroid that is assigned to. Each data point is assigned to the cluster centroid that is nearest to it. Once each datapoint has been assigned to its nearest centriod, the centroids are recalculated as the mean of all the data points in that centroid. All the data points are then reassigned to the centroids since their positions have changed. This process is repeated until the cluster centroids no longer change values substantially when being recalculated.

2.3 K-Means Formal Definition

K-Means is an unsupervised learning algorithm that clusters data points into discrete groups. Let $\mathbf{S}$ be a finite set of vectors, i.e., $\mathbf{S} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_\lambda\}$ where $\mathbf{x}_i \in \mathbb{R}^n$ is a finite vector, i.e., $\mathbf{x}_i = (x_{i_1}, x_{i_2}, \ldots, x_{i_n})$. The algorithm first requires $k \in \mathbb{N}^*$ as input, where $\mathbb{N}^*$ is a set of all positive whole numbers excluding 0, and then instantiates a set of clusters $\mathcal{C} = \{\mathbf{C}_1, \mathbf{C}_2, \ldots, \mathbf{C}_k\}$. Once $\mathcal{C}$ is created, the algorithm instantiates a set of means $\mathbf{M} = \{\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_k\}$, where $\mathbf{m}_i \in \mathbb{R}^n$ is initialized randomly or with an algorithm such as k-means++. For each cluster $\mathbf{C}_i \in \mathcal{C}$, $\mathbf{m}_i \in \mathbf{M}$ is considered to be its centroid. After the initialization of means, each vector $\mathbf{x}_i \in \mathbf{S}$ is assigned to a cluster $\mathbf{C}_j \in \mathcal{C}$ based on the minimum euclidean distance between $\mathbf{x}_i$ and $\mathbf{m}_j$. More specifically, for each mean $\mathbf{m}_j \in \mathbf{M}$, the euclidean distance between $\mathbf{x}_i$ and $\mathbf{m}_j$ defined by equation 2.2 is computed. Then, $\mathbf{x}_i$ is assigned to the cluster $\mathbf{C}_j$ whose centroid $\mathbf{m}_j$ is the closest to $\mathbf{x}_i$.

$$\text{dist}(\mathbf{x}_i, \mathbf{m}_j) = \|\mathbf{x}_i - \mathbf{m}_j\|^2 \tag{2.2}$$

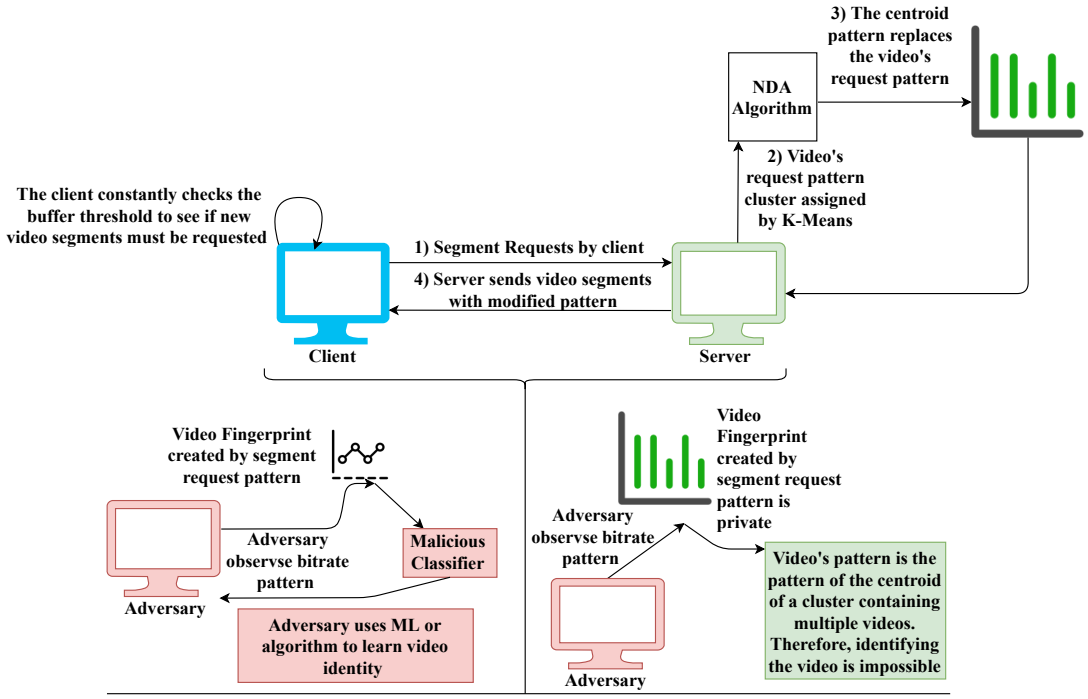


Figure 2.1: A Conceptual Overview of the Proposed Scheme *NDA*

After the assignment phase, each cluster $\mathbf{C}_i \in \mathcal{C}$ should be a subset of $\mathbf{S}$ and given another cluster $\mathbf{C}_j$, with $i \neq j$, $\mathbf{C}_i \cap \mathbf{C}_j = \varnothing$.

Finally, $\mathbf{m}_j$ the centroid of $\mathbf{C}_j$ is updated following equation 2.3:

$$\mathbf{m}_j = \frac{1}{|\mathbf{C}_j|} \sum_{i=1}^{|\mathbf{C}_j|} \mathbf{x}_i^{(\mathbf{C}_j)} \tag{2.3}$$

where $\mathbf{x}_i^{(\mathbf{C}_j)}$ represents the vector in $\mathbf{C}_j$ at position i.

The assignment and centroid update process is repeated until the value of each centroid remain constant.

2.4 Convolutional Neural Network

Deep Neural Networks (DNNs) are one of the most widely used machine learning algorithms largely because of their ability to learn high level, complex features. Deep neural networks are machine learning algorithms that operate using a series of layers, the final

layer being the layer that outputs classification and the first layer being the data that is being input into the neural network. The layers between the input layer and the classification layer are known as hidden layers. Each network varies with the amount of hidden layers it contains. Within each hidden layer are neurons. Each neuron contains an activation function, which will control whether the neuron ”fires”, to use the common analogy of the human brain. The input and output of each neuron is controlled by weights. These weight values are adjusted over time to help the neural network grow more accurate. This happens by the process of back-propagation, which measures the output of the network against the real value, and then adjusts weights accordingly.

As a result of their effectiveness, they became a foundation for many variant neural networks, such as Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM), and Convolutional Neural Networks (CNN). Each of these variants have their own strengths, for example, RNNs are frequently used in natural language processing. A Convolutional Neural Network is an augmentation of the Neural Network with a number of unique components that make it exceptionally good for dealing with multi-dimensional inputs and data that hold strong spatial correlations, in the case of video streaming, large bursts of bit data occur at a certain time in the video. CNNs were effectively applied in the attack scenario put forward by Schuster et al. (?, ?).

CNNs perform multiple transformations to their data that make them unique. One of these is a convolutional layer, that convolves the data. The output of the convolutional layer is a dot product computed between a filter and the input. The number of filters and size of filter can be altered so that the number of channels can increase after a convolutional layer. The purpose of the convolutional layer is to learn key features in the data such as lines, edges, textures etc. in the case of image classification.

Another augmentation of a CNN is the pooling layer. The purpose of the pooling layer is to reduce data dimensionality while preserving key features. Pooling dimensions $n \times n$ are set that are smaller than the input space, a subset of the feature map is then looked

at through this pooling window of n × n dimensions. The pooling layer then takes only one value from this window, it can be the average of all the values in the window, or the maximum etc., but the maximum is the most common value to use. The pooling layer then outputs the maximum value of this n × n section of the feature map and moves a given number of spaces. The pooling window can only move one element in the feature map array or move multiple, the value that controls the amount of elements moved is called the stride. The pooling layer does not have weights or bias, the only purpose of the pooling layer is to reduce size while preserving key features, which is why max pooling is more effective than average pooling, because it will record only the most prominent feature in each section of the feature map.

Finally, the feature map is flattened to a 1 dimensional vector (this is not required, and some notable CNN models don't do this) and there is a fully connected layer before the classification layer to promote non-linear learning.

2.5 Differential Privacy

Differential privacy is a technique for privacy protection. It relies on the addition of noise to data in order to keep individual user data private. The first paper to formally outline what is know as ε-differential privacy was published in 2006 (Dwork, McSherry, Nissim, & Smith, 2006). In layman's terms, a scheme is determined to be differentially private if an observer cannot distinguish between the presence or absence of one single individual's data in the database. Formally, this definition is given by:

$$Pr[A(D_1) \in S] \leq exp(\varepsilon) \times Pr[A(D_2) \in S] \tag{2.4}$$

In this formal definition, Pr stands for probability, A represents a differential privacy analysis mechanism on a database, D_1 or D_2 represent two databases that differ by only one data point, and ε represents the metric that controls the amount of privacy.. The lower

the value of ε, the more noise is added, the more privacy is added. In layman's terms, we could say that a system is differentially private if the probability of output in the space S from mechanism A on D_1, when divided by the same value for D_2 is less than or equal to e^{ε}. Sometimes the definition is alternatively written as:

$$\frac{Pr[A(D_1) \in S]}{Pr[A(D_2) \in S]} \leq exp(\varepsilon) \tag{2.5}$$

Differential privacy works by adding noise within a range that can be controlled. The most common way to add noise to a database is to add the noise within a distribution that is centered on the data-set. The most common distribution for adding noise to a database is the Laplacian distribution, but the Gaussian distribution can be used as well.

For the purpose of comparison, we implemented two differential privacy mechanisms in this paper. The two differential privacy mechanisms we implemented were adjusted for time series data. The two methods we used were $d*-privacy$ (Xiao et al., 2015) and FPA_k (Rastogi & Nath, 2010).

2.5.1 Fourier Perturbation Algorithm (FPA_k)

FPA_k relies on a Discrete Fourier Transformation (DFT). For this algorithm, we consider $Q = (Q[1], \ldots, Q[n])$ to be a real or complex valued sequence of length n. In our case, it is a single video's data. In a DFT, this sequence Q is tranformed into $F = (F[1], \ldots, F[n])$ where:

$$F[j] = \sum_{i=1}^{n} exp(\frac{2\pi\sqrt{-1}}{n} ij) Q[i] \tag{2.6}$$

$F[j]$ represents the j-th Fourier coefficient of the DFT(Q). After a DTF is performed on a data sequence, an inverse DFT (IDFT) can be performed. IDFTs have the property $IDFT(DFT(Q)) = Q$. If we consider a sequence $P = (P[1], \ldots, P[n])$ with complex values, then an IDFT will convert this sequence into another complex-valued sequence

$R = (R[1], \ldots, R[n])$ where:

$$R[j] = \frac{1}{n} \sum_{i=1}^{n} exp(\frac{2\pi\sqrt{-1}}{n} ij) P[i] \tag{2.7}$$

Let Lap(λ) denote a random vareiable drawn from the Laplacian distribution with scale λ and location $\mu = 0$. Suppose the inputs of the FPA_k algirhtnm are Q, λ, and k. The FPA_k algorithm is described as:

- Keep the first k Fourier coeffiecients $F[1], \ldots, F[k]$ after computing DFT(Q).
- Compute $\tilde{F} = F[i] + Lap(\lambda)$ for $i = 1, \ldots, k$.
- Return $\tilde{Q} = IDFT(PAD^n([\tilde{F}[1], \ldots, \tilde{F}[k]]))$, where $PAD^n([\tilde{F}[1], \ldots, \tilde{F}[k]])$ denotes the sequence of length n obtained by appending $n-k$ zeroes to $\tilde{F}[1], \ldots, \tilde{F}[k]$.

Rastogi (Rastogi & Nath, 2010) that FPA_k is ε-differentially private for $\lambda = \sqrt{k}\Delta_2(\mathbb{Q})/\varepsilon$, where $\Delta_2(\mathbb{Q})$ denotes the L2 sensitivity of a set of Qs. Formally, $\Delta_2(\mathbb{Q})$ is the smallest number such that for all $Q, Q' \in \mathbb{Q}, |Q - Q'|_2 \leq \Delta_2(\mathbb{Q})$.

2.5.2 d*-privacy

Xiao (Xiao et al., 2015) leveraged d-privacy with a distance metric $d*$ on a one-dimensional time series. Let x and x' denote two time series. The $d*$ was defined as:

$$d*(x, x') = \sum_{i \leq 1} |(x[i] - x[i-1]) - (x'[i] - x'[i-1])| \tag{2.8}$$

To achieve $d*$-privacy, Xiao (Xiao et al., 2015) extended a mechanism from Chan (Chan, Shi, & Song, 2011) to implement a $d*$-privacy mechanism as follows: Let $\mathbb{N}$ donte the natural numbers and $D(i) \in \mathbb{N}$ denote the largest power of two that divides i; i.e., $D(i) = 2^j$ if and only if $2^j | i$ and $2^{j+1} \not| i$. Note that $i = D(i)$ if an only if i is a power of two. The

mechanism A computes a noised value $\tilde{x}[i]$ that is used in place of $x[i]$ using the recurrence:

$$\tilde{x}[i] = \tilde{x}[G(i)] + (x[i] - x[G(i)]) + r_i \tag{2.9}$$

where $x[0] = \tilde{x}[0] = 0$, and:

$$G(i) = \begin{cases} 0 & \text{if } i = 1 \\ i/2 & \text{if } i = D(i) \geq 2 \\ i - D(i) & \text{if } i = D(i) \end{cases} \tag{2.10}$$

$$r_i \sim \begin{cases} Lap(\frac{1}{\varepsilon}) & \text{if } i = D(i) \\ Lap(\frac{\lfloor log_2 i \rfloor}{\varepsilon} & \text{otherwise} \end{cases} \tag{2.11}$$

It was proven by Xiao (Xiao et al., 2015) that the algorithm in Equations 2.9,2.10, and 2.11 is $(d*, 2\varepsilon)$-private and $(l_1, 4\varepsilon)$-private.

Chapter 3

Problem Description

3.1 Traffic Analysis Attack

The traffic analysis attack against MPEG-DASH video streaming relies on side channel information to identify the video a user is streaming. While video streaming, a client requests video segments from the server at regular intervals. The video segments themselves are encrypted, but the meta-data including packet size and arrival times are visible at the application layer to any adversary on the network (Schuster et al., 2017). The bit-rate data seen by the adversary can be used to determine whether or not a user is streaming a specific video selected by the adversary. Multiple approaches have been taken to use this data for malicious purposes, both algorithmic and machine learning based. The algorithmic approaches (Gu et al., 2019; Reed & Klimkowski, 2016) seek to measure similarity between the users bit-rate data and the adversary's pre-recorded bit-rate data for a specific video. The machine learning approaches (Schuster et al., 2017; Dubin et al., 2017) seek to predict whether or not a user is watching the video selected by the adversary. Schuster *et al.* introduced a traffic analysis attack based on a CNN and extended it to work in a web browser, it is executed by JavaScript code that saturates a victim's connection to a server and then measures the traffic changes (Schuster et al., 2017). More information about the attacks

and their implementation is included in Section 8.

However this attack is implemented, whether by machine learning or an algorithmic approach, the data it relies on is the same. The vital information being leaked is the size of the packets and the times of their delivery, which allow an adversary to observe the rate at which bits are sent, or the bit-rate of the video stream. The un-obfuscated graph in Figure 4.1 is a graphical representation of the format of this bit-rate data. Because of MPEG-DASH and VBR, these bit-rate patterns are a unique fingerprint for at least 20% of videos when analyzed theoretically (Schuster et al., 2017), though all implementations of this attack show accuracy values above 90% for video identification.

For this attack, we make two assumptions. First, we assume a polynomial time adversary, that is, an adversary restricted to practical means of attack. Second, we assume that the adversary is external, and cannot be executing their attack from the server side.

3.2 Problem Definition

Creating a defense mechanism for this attack is in theory straightforward. The video request pattern, seen as bit-rate by the adversary, must be changed so that an adversary can no longer use this information to compromise user privacy. In practice, there are more considerations, primarily computational efficiency. Video streaming is a computationally expensive process, and if a request pattern is obfuscated too much it will cause video lag or video buffering because not enough data is being sent, or conversely, because excess data is being downloaded. To the best of our knowledge at the time of this writing, only one defense strategy has been proposed, by Zhang *et al.* (Zhang et al., 2019). It leverages differential privacy and works by setting a proxy between the client and server in the form of a browser extension. The extension perturbs the video segment request pattern using differential privacy. Differential privacy adds noise to data, in the case of video streaming, this noise changes the time intervals of the requests from the client and the amount of data

requested by the client. The defense scheme proposed by Zhang *et al.* (Zhang et al., 2019) leverages two differential privacy methods for obfuscation, $d^* - privacy$ and FPA_k.

While the strategy of video request pattern obfuscation with differential privacy successfully defended against the CNN based attack proposed by Schuster *et al.* (Schuster et al., 2017), there was computational overhead incurred by the defense because of the use of differential privacy, which always trades utility for privacy. Because of this, and the need for scalability in the field of video streaming, we sought to create a more efficient defense solution.

In our attempt to define constraints for a more efficient, private solution, we considered that a defensive scheme should improve with more available data, growing more robust over time. Additionally, we assert that the solution must be scalable, considering the scale of the video streaming industry, the number of users who stream on computationally constrained mobile devices, and the computational cost of video streaming.

3.3 Privacy Definition

For our scheme we created a formal privacy definition based on the type of privacy provided by our scheme. A formal privacy definition gives our scheme scope beyond just the experimentation done. The privacy of our scheme is given by the mapping of multiple inputs to a single output. When we obfuscate a cluster of videos, all the videos in the cluster are obfuscated to a single output value. This means that if there are 10 videos in a cluster, if the adversary is only able to observe the streaming after obfuscation, the adversary has a 10% chance of guessing the identity of the video. If the adversary does guess the title of the video correctly, they have no way to verify it. Therefore, our privacy definition is based off of the number of videos in a cluster. The merit of providing privacy in this way is that as the scale of the scheme increase, the privacy does too. We give our privacy definition as follows:

We define n as the total number of videos in the data-set that are clustered, G as the guess of the adversary, S as an obfuscated video capture, V as the true identity of the video, $Pr[G(S) = V]$ is the probability of the correctness of the attacker's guess, and ε is a negligible probability. We assume the adversary has full knowledge of the bit-rate of a single centroid. The probability that the attacker is correctly able to guess the title given by the centroid can be expressed by:

$$Pr[G(S) = V] \leq \frac{1}{n} + \varepsilon \tag{3.1}$$

If multiple clusters are created, then n represents the smallest cluster size, i.e., if there are 3 clusters, and the smallest cluster has only 5 videos, then $n = 5$. Though in experimentation the privacy shown is much greater than this.

3.4 Distance Privacy Definition

We sought to create a distance privacy definition to give a more in depth view of the privacy being provided. Drawing from the attack paper by Schuester *et al.* (Schuster et al., 2017), we used the L_1 norm as the basis of our privacy definition. The L_1 norm can be defined for two vectors $\mathbf{a}, \mathbf{b} \in \mathbb{R}^n$ by the following equation:

$$L_1(\mathbf{a}, \mathbf{b}) = \|\mathbf{a} - \mathbf{b}\|_1 = \sum_{i=1}^{n} |\mathbf{a}[i] - \mathbf{b}[i]| \tag{3.2}$$

For this privacy definition, we consider two vectors $\mathbf{x}$ and $\mathbf{y}$. Let the vector $\mathbf{x}$ represent video byte data recorded at a constant interval $t = 0.25$ seconds over a time space T. Let $\mathbf{y}$ represent the vector $\mathbf{x}$ after obfuscation. In the paper by Schuester *et al.* (Schuster et al., 2017), the adversary was successfully able to identify a video if the L_1 norm between the recorded data $\mathbf{x}$ and the attacker's fingerprint $\mathbf{y}$ was less than 3,500,000 bytes. To make a robust privacy model, we reduced this threshold to 2,200,000 bytes so that videos must

have an increased level of privacy, making the scenario more favorable to the attacker.

We also used this privacy definition to compare our proposed scheme *NDA* to differential privacy (Zhang et al., 2019). For our own scheme *NDA*, privacy is two-fold. There is privacy given by the obfuscation of the original video, and there is privacy given by belonging to a cluster with a high number of videos. If a cluster has 10 videos, guessing at random the adversary has a 10% chance of guessing the correct video even if the adversary has full knowledge of which videos are in the cluster. Therefore, for *NDA*, we multiply the L_1 norm by the number of items in the cluster to account for the extra privacy provided by being included in a cluster with an increasingly large number of data. Additionally, this will account for the degradation of privacy that comes when the number of clusters increases, causing the number of videos per cluster to decrease. Letting $\mathbf{C}$ be a cluster, our scheme's privacy is shown by $(|\mathbf{C}| \times \|\mathbf{x}-\mathbf{y}\|_1) \leq 2,200,000$. This privacy threshold of 2,200,000 bytes is further validated with the accuracy levels shown later in the experimentation. For differential privacy, we used only the L_1 norm as the privacy measure, represented by $\|\mathbf{x}-\mathbf{y}\|_1 \leq 2,200,000$, because all the privacy given by differentially private solutions comes from noise added. It is therefore logical to conclude that a measurement of the distance between two vectors because of added noise in order to preserve privacy will give a clear view of the level of privacy provided.

Chapter 4

Proposed Scheme 1: NDA

4.1 Metrics

For evaluation of the video performance after the implementation of our algorithm, we used the two metrics defined by *Zhang et al.*(Zhang et al., 2019), waste and deficit. Both metrics are defined in relation to the bit-rate pattern of the original video. Deficit can be defined as the maximum difference between what amount of video is being downloaded in the obfuscated request pattern, and what amount of video should be downloaded. Waste can be defined as the opposite, the amount of extra video that is being downloaded that doesn't need to be. Let $\mathbf{a} \in \mathbb{R}^n$ be a vector that represents the original un-obfuscated video pattern and let $\mathbf{b} \in \mathbb{R}^n$ be a vector that represents the obfuscated video pattern.

$$waste = \max_{1 \leq i \leq n} \{max(\mathbf{b}[i] - \mathbf{a}[i], 0)\} \tag{4.1}$$

$$deficit = \max_{1 \leq i \leq n} \{max(\mathbf{a}[i] - \mathbf{b}[i], 0)\} \tag{4.2}$$

4.2 Proposed Scheme Overview

Our proposed scheme, "No Data are Alone (*NDA*)", seeks to find an efficient and effective way to obfuscate video requests from a client. Figure 1 depicts our overall scheme in detail. In our proposed scheme, the server has a database of all video segment request patterns from which a random subset will be selected to fit a K-Means algorithm on. Let a video request pattern be defined as a vector $\mathbf{x} \in \mathbb{R}^n$, where $\mathbf{x}[i]$ represents one video segment sent from the server to the client. Let $\mathbf{S} = (\mathbf{x}_1, \ldots, \mathbf{x}_m)$ be the set of video request patterns. In our scheme, the server has knowledge of $\mathbf{S}$ and its contents. When a client selects a video for which its request pattern is $\mathbf{x}_i \in \mathbf{S}$, this video request pattern $\mathbf{x}_i$ is altered by our algorithm *NDA* so that its value is now $\mathbf{y}_i$.

As a client streams a video, the client will send requests to the server. Each request is filled with a segment of video that can be defined as $\mathbf{x}_i[j]$ from the vector $\mathbf{x}_i$. In an unaltered system, the vector $\mathbf{x}_i$ is defined progressively by the size of data sent from the server to the client, with the size of each video request x_i (in bytes) being dependent on the content of the video clip, the desired streaming quality, and the quality of the network the client is streaming from. Under our proposed scheme, the requests are not filled according to the request of the client, but according to the vector $\mathbf{y}_i$.

In order to preserve video streaming quality, the vector $\mathbf{y}_i$ is defined by our algorithm and minimizes the two metrics defined in Section 4.1.

4.3 Proposed *NDA* Algorithm

Our algorithm is an augmentation of K-Means clustering. We use K-Means clustering instead of other clustering algorithms because K-Means clustering provides a centroid. This centroid is the average of all values in the cluster, and is crucial to our scheme. With another clustering algorithm, we would need to compute the value of the centroid ourselves. One of the advantages to using K-Means clustering is that initialization is different each

time the model is fit, and therefore cluster distribution is also different with each fitting. The implication of this is that even if the attacker knows the full set of videos, and performs his own clustering, he will not receive the same cluster distributions. So if the adversary determines that the target video V is in his cluster $Cluster$ 1, this will not necessarily be true in the defensive schemes cluster distribution.

In our scheme, first, we apply K-Means clustering to the set of videos $\mathbf{S}$. In a naive approach, the centroid $\mathbf{m}_j$ of a cluster $\mathbf{C}_j$, which is defined as a vector $\mathbf{m}_j = (x_1, x_2, \ldots, x_n)$ and is calculated by equation 2.3, can then serve as an obfuscated pattern for each video $\mathbf{x}_i \in \mathbf{C}_j$. Theoretically, since there are multiple videos in each cluster, the adversary cannot distinguish between them if they are all streamed with the same (centroid) pattern. In practice, the naive approach encounters problems and this theory does not hold (see Figure 6.2).

While clustering data, it is inevitable that some data points $\mathbf{x}$ will be alone in a cluster $\mathbf{C}$. When this is the case, the mean $\mathbf{m}$ calculated will be equal to the data point $\mathbf{x}$ so that this instance can be shown by considering:

$$\mathbf{m} = \frac{1}{|\mathbf{C}|} \sum_{i=1}^{|\mathbf{C}|} \mathbf{x}_i^{(\mathbf{C})} = \mathbf{x} \tag{4.3}$$

where $|\mathbf{C}| = 1$.

In this case, implementation of the naive algorithm would result in the obfuscated pattern $\mathbf{y}_i$ being equal to $\mathbf{x}_i$, and no privacy would be provided. To combat this, we developed No Data are Alone (*NDA*).

The assumptions made by our algorithm are as follows: a user will request video segments $(x_{i_1}, x_{i_2}, \ldots, x_{i_n}) \in \mathbf{x}_i$ for some video $\mathbf{x}_i$, the server will fulfill these requests with an obfuscated pattern $\mathbf{x}_i$. The server has a database of videos $\mathbf{S}$ and an *NDA* model N that is fit on a random subset of these videos. Whenever a video $\mathbf{x}_i$ is requested by the user, the server must compute $\mathbf{y}_i$. This computation is the same as the assignment step in K-Means. Instead of directly returning this result as $\mathbf{y}_i$, our algorithm will check the value of $|\mathbf{C}_j|$. If this value is 1 (meaning the data is alone in a cluster), our algorithm performs a new

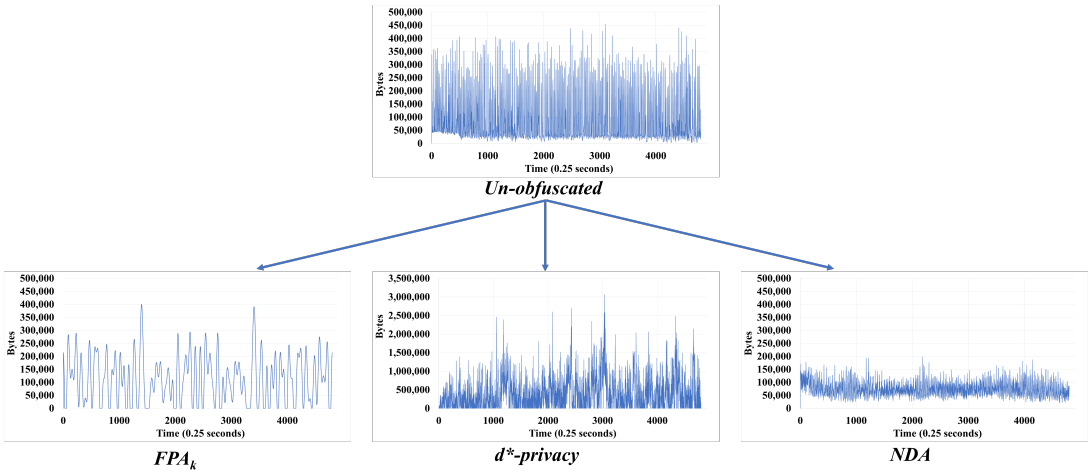


Figure 4.1: Obfuscation Pattern Comparison

assignment.

Our algorithm's reassignment step is based on the minimization of $waste + deficit$ instead of Euclidean distance, and requires $|\mathbf{C}_j| > 1$ so that the cluster assignment $\mathbf{C}_j$ of a video $\mathbf{x}_i$ will have not be alone in a cluster, ensuring privacy, and an obfuscated pattern $\mathbf{y}_i$ that minimizes waste and deficit when compared to all other cluster patters $\mathbf{M} = \{\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_k\}$. Our algorithm can be represented as follows:

$$\begin{aligned} \mathbf{C}_j := argmin(& \max_{1 \le i \le n} \{ \max_{j \in k} \left(\mathbf{m}_j[i] - \mathbf{x}_i[i], 0 \right) \} \\ & + \max_{1 \le i \le n} \{ \max_{j \in k} \left(\mathbf{x}_i[i] - \mathbf{m}_j[i], 0 \right) \}) \end{aligned} \tag{4.4}$$

For Algorithm 1 let $\mathbf{M}$ be a list of the cluster centroids, where $\mathbf{m}_j \in \mathbf{M}$ is the centroid of a cluster $\mathbf{C}_j$. $\mathbf{x}$ represent the video segment to be made private, and $\mathbf{y}$ is the video segment after obfuscation. All cluster centroids are evaluated to determine which cluster assignment $\mathbf{C}_j$ produces the lowest value of waste + deficit.

Algorithm 1: No Data Are Alone (NDA)

Input : M, x

let min_waste = 0
let min_deficit = 0
let $\mathbf{y} \in \mathbb{R}^n$
for *each centroid* $\mathbf{m}$ *in* $\mathbf{M}$ **do**
    let $\mathbf{C}$ be the cluster corresponding to $\mathbf{m}$
    if $|\mathbf{C}| = 1$ **then**
        remove $\mathbf{m}$ from $\mathbf{M}$
    end
end
for *each centroid* $\mathbf{m}$ *in M* **do**
    $waste = \max_{1 \leq i \leq n}\{max(\mathbf{m}[i] - \mathbf{x}[i], 0)\}$
    $deficit = \max_{1 \leq i \leq n}\{max(\mathbf{x}[i] - \mathbf{m}[i], 0)\}$
    if $waste < min\_waste$ *and* $deficit < min\_deficit$ **then**
        $min\_waste = waste$
        $min\_deficit = deficit$
        $\mathbf{y} = \mathbf{m}$
    end
end
return y

4.4 Cluster Recreation Probability

Our proposed schemes maps multiple inputs to one output. Multiple videos are in any given cluster, and these videos will all have their patterns obfuscated to the same cluster centroid pattern. When mapping multiple videos to one centroid, the probability of guessing which video is mapped to the output is dependent on the number of videos in the cluster. If there are 10 videos in the cluster, the probability of guessing based off of the output would be 10%. A concern for a scheme that provides privacy in this way is the recreation of the same mapping. In the case of our scheme, *NDA*, the adversary would have to produce the same cluster distribution. In our scheme, we performed clustering on 40 videos (though this number could greatly increase in real world implementation).

To consider the privacy given my our scheme more fully, we consider the possibility of the adversary recreating the same cluster distribution used by our defensive scheme. We give the adversary full knowledge of all 40 videos that were used to perform the *NDA*

algorithm. Using this knowledge, the adversary is allowed to perform his own clustering to attempt to obtain the same distribution as our proposed scheme. A cluster distribution can be defined as which videos belong to which clusters i.e., there are 4 videos in cluster 1, 12 videos in cluster 2, etc. If the adversary is able to obtain a distribution with the same videos in the same cluster as the defensive scheme distribution it would be a breach of privacy.

For the cluster initialization algorithm in our scheme, we use K-means++ (Arthur & Vassilvitskii, 2006). This algorithm will determine the probability of obtaining the same cluster distribution twice. This algorithm randomly selects a data-point as a starting cluster centroid, then initializes the rest of the cluster centroids with probabilities proportional distance from the chosen data-point i.e., a cluster with a distance closer to the chosen centroid has a lower probability of being chosen as the next centroid, while a data-point the furthest away from the initial cluster centroid has the highest probability of being chosen. This means that with K-means++, there are 40 possible initial data-points to choose in our data-set. This means the adversary has a at most a $\frac{1}{40}$ or 2.5% chance of producing the same initial cluster centroid. The probability of maintaining the same cluster distribution degrades with each subsequent assignment. Additionally, the adversary has no way of knowing if he has successfully produced the same cluster.

4.5 Various Clustering Algorithms

While doing experimentation on K-Means clustering, we also performed clustering with two other algorithms, Agglomerative clustering and DB-SCAN clustering. Agglomerative clustering and DB-SCAN are two popular clustering algorithms. When performing clustering, we noticed the exact same pattern in each of the clustering algorithms. Each algorithm provided very lopsided clusters, most of the videos were grouped into a single cluster with only a few videos falling into other clusters. We fit the models multiple times but the result was always the same. Since the output for all three clustering algorithms was nearly iden-

tical, we decided to use K-Means clustering. Additionally, our schemes need the cluster centroid in order to provide privacy. K-Means provides this as part of its implementation, but it would have to be calculated manually for another clustering algorithm. This would introduce a small amount of computational overhead because it would have to be implemented manually instead of using a package optimized for it as we did with K-Means, and since the distributions were very similar across different algorithms, the centroids would be as well.

Chapter 5

Implementation and Simulation

5.1 Data Collection

Data collection was automated using tshark by Wireshark[1] and Selenium[2]. We collected data from YouTube, and only videos of 20+ minutes were captured, ad content was filtered out, and video quality was kept constant (720p). We recorded the server to client bit-rate of each video in segments of 0.25 seconds. We collected a data-set of 41 different hand selected videos. The bit-rate data of each these 41 videos was collected for 100 captures each, and each capture only lasted for exactly 20 minutes. With a 20 minute long capture that captured data every 0.25 seconds, each video capture had 4800 data-points so that a single video capture could be represented as $\mathbf{x}_i = (x_{i_1}, x_{i_2}, \ldots, x_{i_{4800}})$. With 100 captures for 41 videos, we ended up with a data-set of 4100 samples, 100 per video, with 4800 data-points for each video. This data was used to implement our defense algorithm and train our CNNs. We also collected 1000 traces of 20 minute long videos, each unique. These videos were split into 10 categories, Boxing, Soccer, Basketball, Football, League of Legends, Fortnite, Makeup Tutorials, Vlogs, Symphony Performances, and Ted Talks. These videos weren't used during experimentation, but are used in Figure 6.1 to elaborate on our scheme.

[1] www.wireshark.org
[2] www.selenium.dev

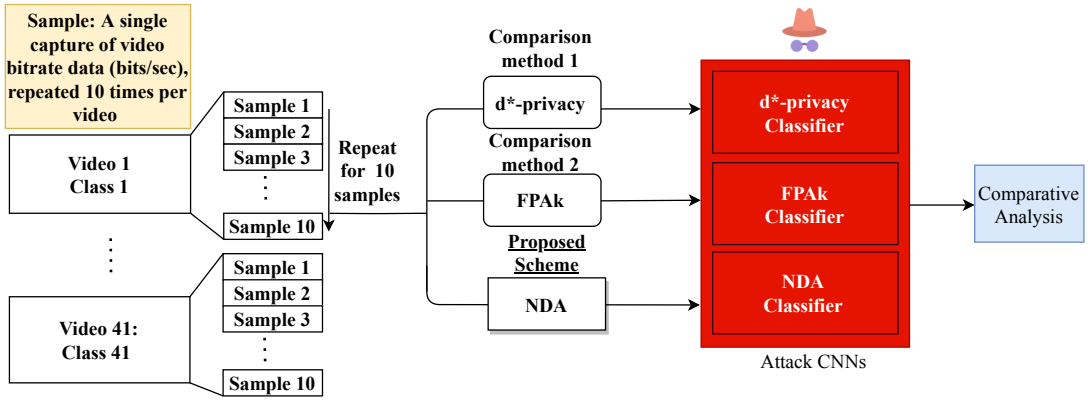


Figure 5.1: An Overview of the Evaluation Methods

5.2 Comparison of Defense Schemes

To evaluate the performance of our scheme *NDA* compared to the proposed scheme of Zhang *et al.* (Zhang et al., 2019), we implemented $d^* - privacy$ and FPA_k exactly the same as Zhang *et al.* with one modification. The value for k in the paper by Zhang *et al.* was set a 10, and their video length was 720. Our video length was 4800, so accordingly we increased k to 67.

The un-obfuscated graph in Figure 4.1 is a graph of the data exploited by this attack, bit-rate over time. The bursty nature of video streaming can be seen here, the graph continues at a low number of bytes, then a large spike (burst) in the graph occurs when a client's request is filled. A graph of our defensive method *NDA* and graphs of each differential privacy method are shown to add a deeper analysis of each method, beyond just the waste and deficit measure in Section 6.

The *NDA* graph in the bottom right of Figure 4.1 is a graphical representation of the obfuscation that our proposed scheme creates. Our scheme provides obfuscation by computing an average of many video patterns like the un-obfuscated graph at the top. This average is the cluster centroid. Because the centroid is the average of multiple videos, *NDA* has slightly smaller bursts than the un-obfuscated pattern, but still retains the bursty nature. In the experimentation from which this graph was derived, 36% of videos were

assigned to this cluster. In our scheme, the 36% of videos assigned to this cluster will be obfuscated with the pattern shown in the *NDA* graph in Figure 4.1.

The FPA_k graph in Figure 4.1 is a representation of obfuscation by FPA_k, which relies on a Fast Fourier Transformation, addition of laplacian noise, and subsequent Inverse Fast Fourier Transformation for its obfuscation. It can be seen that this video doesn't exhibit the bursty nature of video streaming, but instead has more gradual fluctuations, which could lead to video lag because of prolonged periods without requesting new video segments.

The $d^* - privacy$ graph in Figure 4.1 is a graphical representation of $d^* - privacy$, which adds simple laplacian noise to time series data. This method adds the most noise, the range of bytes for the original un-obfuscated video stays mostly within the 50,000-300,000 range, but the $d^* - privacy$ method has a large number of points in the 1,000,000 bytes range, which would cause an excess downloading of video data. This result agrees with the waste and deficit measurements given in Section 6.

5.3 Attack Classifiers

To test our proposed algorithm, we implemented the CNN created by Schuster *et al.* (Schuster et al., 2017), with a few minor modifications to the architecture. We did this to accommodate our data vectors, which were significantly longer than the ones used by Schuster *et al.* We used a filter size of 32 with a kernel size of 3 and a pooling size of 2 instead of 6. We also used the Adadelta optimizer instead of Adam. Additionally, we used z-score normalization and a learning rate of 0.001. These were the only differences. Our classifier has 41 classes, one for each video. We trained this classifier on our full data-set for 80 epochs. The classifier, $Model_A$, had an accuracy of 0.9316 and a false positive rate of 0.0017.

For evaluation we used 10 samples for each video, totalling 410 samples for all 41 videos. The results of accuracy, waste, and deficit for all 410 samples tested were averaged to give a broad view of the performance of each algorithm. Figure 5.1 is a visual

representation of these evaluation methods.

We created three "attack CNNs" and trained one on noised data from our scheme *NDA* and one on noised data from each of the two differential privacy schemes, in an attempt to increase the performance against them. We then tested these defense schemes against our attack CNNs and recorded the waste and deficit for *NDA*, and for the differential privacy schemes with varying epsilon values. Results of the accuracy for each of these schemes can be seen in Table 5.1.

To train against our scheme *NDA*, we used one data-set of videos obfuscated by our scheme and the original un-obfuscated data-set. Instead of training a new classifier, we retrained the original model, $Model_A$. This new attack model, $Model_B$, was trained for 50 epochs, or else significant over-fitting occurred. This model theoretically shouldn't be able to successfully learn to predict our scheme, because of the method our scheme uses to provide privacy. When we fit our algorithm with 4 clusters, all videos can be obfuscated to one of four options. This may result in, for example, 15 of 41 videos all being assigned to the same cluster, and obfuscated with the same pattern. The classifier will be unable to learn any correlation between an obfuscated pattern and a video class, because so many videos from different classes will have the same pattern. The results in Section 6 support this conclusion.

To create the FPA_k attack CNN, $Model_C$, we retrained $Model_A$ on 5 data-sets. We included the original un-obfuscated data-set, then 4 different data-sets of data obfuscated by FPA_k, two with an epsilon values of 15 and two with epsilon values of 25. Different epsilon values will yield classifiers robust to different levels of obfuscation. We chose 15 and 25 to have a well balanced model. We trained the model for 500 epochs. This model required more data and longer training time to become accurate when compared to the $d^* - privacy$, which is unsurprising when you consider FPA_k in Figure 4.1 and the higher level obfuscation when compared to $d^* - privacy$ which added significant noise but retained the bursty pattern. This model had an accuracy of 0.9317 and a false positive rate

Table 5.1: Comparative Analysis: Attack Classifiers

	FPA_k				$d^* - Privacy$				NDA			
Accuracy	ε	W	D	Acc	ε	W	D	Acc	C	W	D	Acc
0 - 10%	0.5	10.45	2.10	0.03	5e-6	29.51	1.68	0.03	4	0.15	2.17	0.07
10 - 20%	5	1.10	2.15	0.16	1.4e-5	10.43	1.42	0.18	N/A	N/A	N/A	N/A
21 - 40%	10	0.57	2.15	0.23	1.8e-5	8.13	1.31	0.33	N/A	N/A	N/A	N/A
40 - 60%	25	0.31	2.16	0.53	2e-5	7.31	1.25	0.44	N/A	N/A	N/A	N/A
$> 60\%$	N/A	N/A	N/A	N/A	1e-4	1.49	0.08	0.93	N/A	N/A	N/A	N/A

ε: Epsilon to achieve the accuracy, W: Waste, D: Deficit, Acc: Exact attack accuracy, C: number of clusters

of 0.0024 on the un-obfuscated data.

To create the $d^* - privacy$ attack model, we trained the $Model_D$ from the original model $Model_A$ with 2 data-sets. We used the original un-obfuscated data and one data-set obfuscated by $d^* - privacy$ with an epsilon value of 0.0007. This model trained for only 50 epochs or else over-fitting occurred.

Furthermore, to show that transfer learning was taking place when we retrained $Model_A$, we reconstructed the original architecture from $Model_A$ and trained it against FPA_k from scratch on the 5 data-sets, however, the accuracy of this model on the un-obfuscated data was significantly lower than retraining the previous model, and lower on data obfuscated by FPA_k that is was trained to classify. This implies that the knowledge about identifying un-obfuscated data successfully transferred from one task (detecting un-obfuscated data) to another (detecting obfuscated data). From this it can be inferred that even after obfuscation with differential privacy, the video pattern retains identifiable features that can be learned.

Chapter 6

Experimental Results

6.1 Privacy Evaluation

To get a broader view of the privacy levels shown by our scheme and the extant scheme, we performed obfuscation on all the videos in our data-set, both with our scheme, *NDA*, and with the two differential privacy schemes used by Zhang *etal*. (Zhang et al., 2019). We then represented the privacy level of each video with a boolean value, either private or non-private based on the distance privacy definition given in Section 3.4. We then divided the total number of non private videos for each scheme by the total number of videos, giving the percentage of non private videos for each scheme. The results are shown in Table 6.1.

These results can also be interpreted inversely, that is, *NDA* with 4 clusters has a 1.4% chance of leaving a video non private, or *NDA* has a 98.6% chance of successfully priva-

Table 6.1: The percentage of non-private videos created by each scheme

Scheme	**Privacy**
NDA 4 Clusters	0.014
NDA 24 Clusters	0.181
$FPA_k\ \varepsilon = 0.5$	0.00023
$FPA_k\ \varepsilon = 5$	0.971
$d^* - privacy\ \varepsilon = 0.000005$	0
$d^* - privacy\ \varepsilon = 0.0001$	0.987

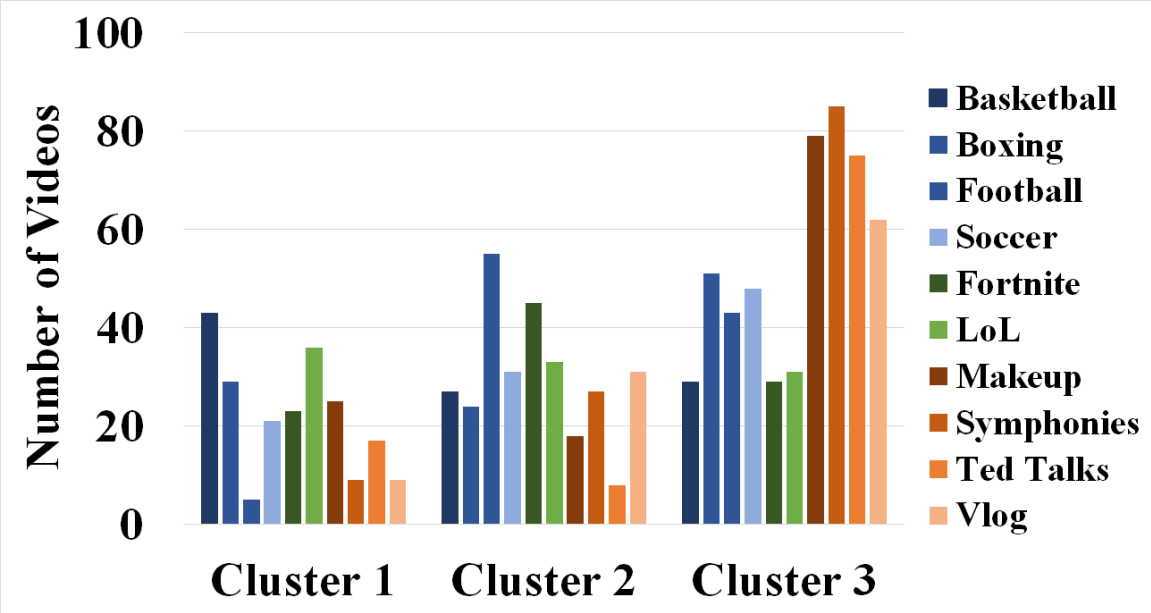


Figure 6.1: Video Category Distribution Across Multiple Clusters

tizing a video. It can also be seen that the privacy of our scheme degrades some as the number of clusters increases, but still remains high. It can be seen here that at an epsilon value of 0.5, FPA_k performs well, however, the waste incurred at this value is high (see Table 5.1). When the epsilon value is increased slightly, the privacy degrades very quickly, more quickly than the attack accuracy using the CNN (see Table 5.1). This calls into question the protection level of this scheme against other attacks. When observing $d* - privacy$, the privacy level looks impressive, but the computational cost of this scheme is very high (see Table 5.1).

6.2 Clusters and Video Types

Figure 6.1 depicts the clustering of 1000 unique videos, that each fall into one of 10 categories. The purpose of this figure is to show the distribution of video category among different clusters. This is important to consider, because it is possible the attacker might try to determine the cluster identity, and then infer the category of the video from the cluster. This graph gives insight to the possibility of invading a user's privacy this way. The

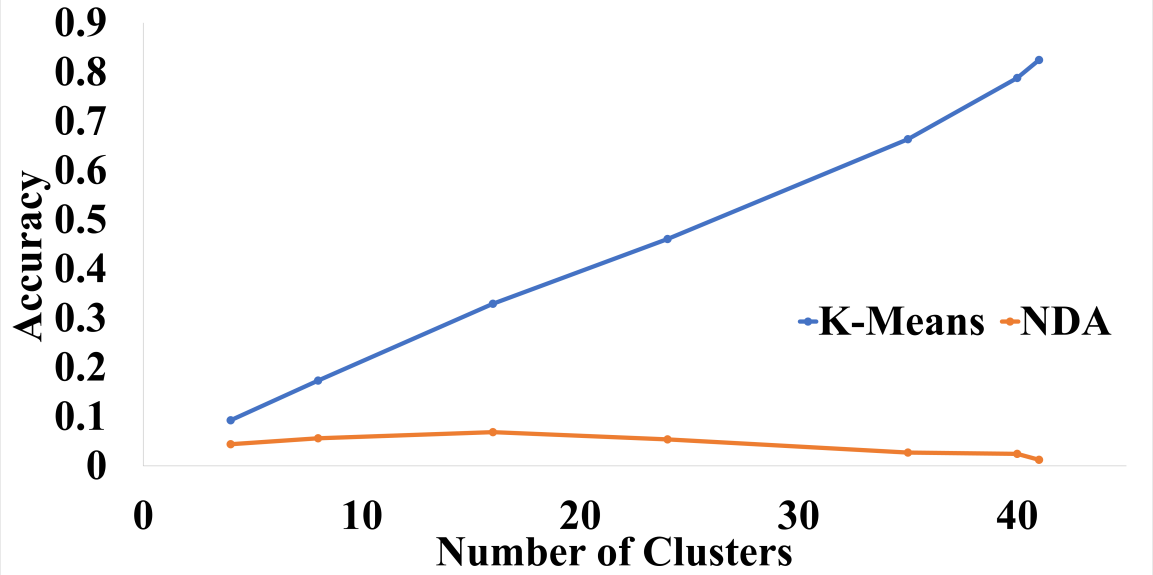


Figure 6.2: Accuracy of the CNN Attack against K-Means and *NDA* across an increasing number of clusters

ten categories in the graph fall into 3 broader categories. Sports videos are shown in blue, video game videos are shown in green, and low action videos are shown in orange. The Y axis of the graph represents the number of videos in a given cluster, i.e. 40 football videos fall into Cluster 1, 20 Football videos fall into Cluster 2 etc. This graph shows a broad distribution of videos even within the same category. League of Legends videos have their highest percentage of videos in cluster 2, along with Basketball. Fortnite, the other group of videos in the video game category has the most videos in cluster 2, along with Boxing videos. From this graph it can be seen that there is no category of videos that dominate a single cluster.

6.3 Traditional K-Means vs. NDA

Figure 6.2 is a measure of accuracy with an increasingly large number of clusters with both traditional K-Means and our proposed scheme *NDA*. This graph displays the need for an altered K-Means algorithm. As the number of clusters increases, the number of videos alone in their own cluster also increases. Data points alone in a cluster have no

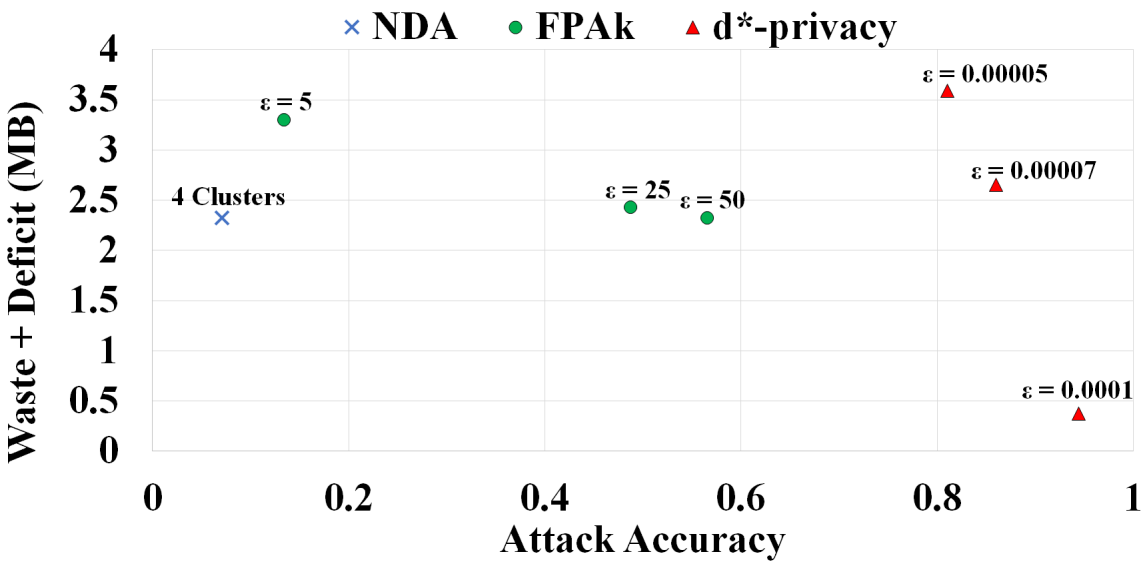


Figure 6.3: Accuracy vs. $waste + deficit$

privacy, because the cluster centroid is equal to the video pattern. This is why as the clusters increase, the number of alone data points increases, and therefore the privacy of the scheme decreases.

This is because with a high number of videos in one cluster, the centroid is an average of all these videos. As the number of alone clusters increases, the cluster reassignment will mean that one video is being assigned another videos pattern, instead of an average of multiple videos. This makes the similarity to the un-obfuscated video pattern decrease compared to a centroid that is the average of many videos. However, this impact is minimal, and the accuracy for NDA never goes higher than 7%.

6.4 Accuracy vs $waste + deficit$

Figure 6.3 is a comparison of the the attack accuracy against each scheme vs $waste + deficit$ (measured in Mega-Bytes) of each scheme; each defensive scheme was tested against the classifier trained to attack it. For $d^* - privacy$ and FPA_k, we considered multiple ε values. Table 5.1 represents a summary of the performance of each defensive scheme when evaluated against the CNN that was trained to attack it, i.e. the performance of *NDA*

against classifier trained on *NDA* obfuscated data. Since we could not achieve an identical accuracy for all three schemes, we set accuracy ranges for the table and found ε-values that made them fall under the ranges.

As the ε values increase, accuracy increases and $waste + deficit$ decrease. This intuitive, a higher ε value will add less noise to a differential privacy scheme, in this case, it means that the obfuscated video is closer to the original and is therefore more likely to be correctly classified, and will also incur less waste and deficit because it is closer to the original.

In every ε case, our scheme outperforms differential privacy when considering both accuracy and $waste + deficit$. Our scheme significantly outperforms $d* - privacy$, and creates 1MB less $waste + deficit$ while having an attack accuracy half as high as FPA_k. The correlation between ε, waste and deficit, and accuracy can also be clearly seen from this graph. Higher ε values will add less noise, resulting in higher attack accuracy and less waste and deficit. We did not include the accuracy for $d^* - privacy$ that would've been equivalent to ours because the waste incurred was extremely high, and we wanted to preserve the scale of the graph (see Table 5.1).

6.5 Attack Accuracy Across Epochs

Figure 6.4 shows the accuracy the attack CNNs of each scheme across multiple epochs of training. In Figure 6.4, Epoch 0 represents the classification accuracy of $Model_A$ before training on noised data. Figure 6.4c depicts the accuracy of the classifier that was trained on both un-obfuscated data and data that was privatized using our algorithm *NDA*. This result shows that over 50 epochs, the classifier doesn't learn anything about about the data, the accuracy remains below 8% the entire training time. This result is predictable because of the privacy preserving format of our proposed scheme. With only 4 clusters, all videos are obfuscated to 1 of 4 patterns. In the testing, 410 samples are taken. All 410 samples

are obfuscated to only 4 videos. With so many samples being obfuscated into one of only four possibilities, no learning can occur.

From figure 6.4a, it can be seen that the FPA_k attack classifier effectively learned all the different epsilon values of FPA_k except for one, when epsilon is 0.5, which adds considerable waste and deficit. The accuracy improvement is expected, because differential privacy always adds noise within a range (controlled by epsilon), so with enough data a classifier can still learn to predict accurately through the obfuscation. We used only 4 obfuscated data-sets to produce this result, but it is logical to conclude accuracy would grow higher against FPA_k if more obfuscated data-sets were used. Additionally, we included one data-set of un-obfuscated data while training the FPA_k attack classifier so that the model did not over-fit and learn only FPA_k.

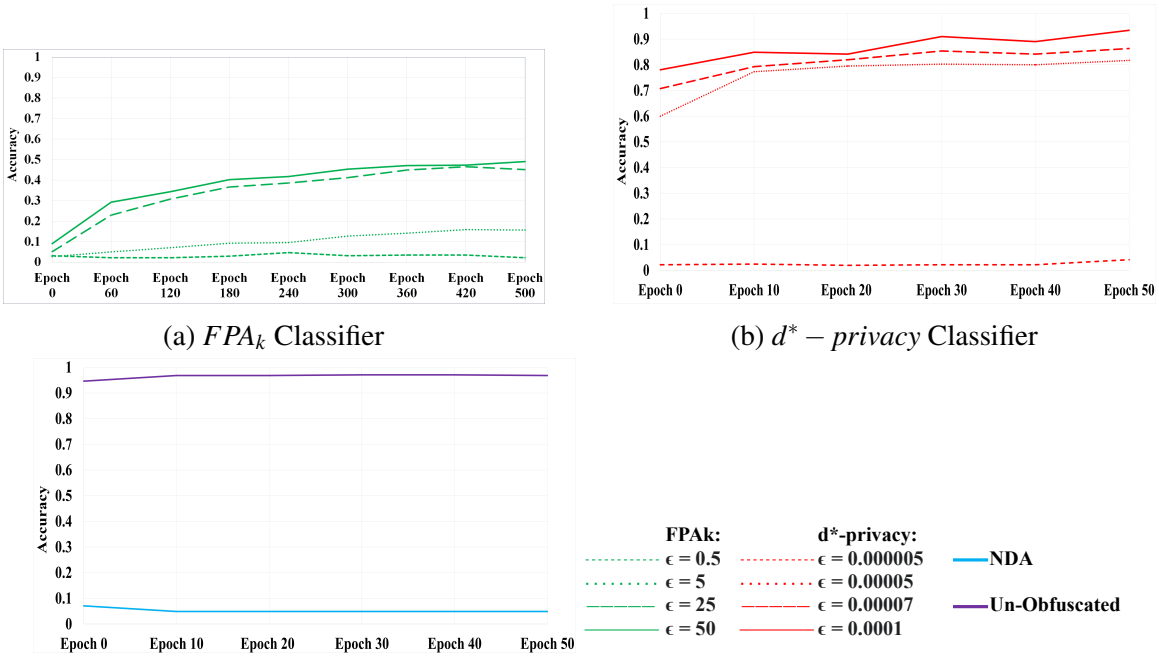
Figure 6.4b depicts the $d^* - privacy$ attack classifier. The accuracy for this scheme can also be seen to increase for all but the highest level of privacy protection, which adds too much waste to be viable.

This is the downside of differential privacy, there is always a trade-off of computational efficiency and privacy, and a scheme that adds too little noise can be overcome by training against the differential privacy scheme, but a scheme that adds too much noise can create too much computational overhead. Our scheme overcomes this trade-off and provides constant high level privacy while being computationally efficient.

6.6 Time Complexity

Because of the real world nature of this problem, and the need for a scalability in the video streaming industry, we examined the computational overhead of both our scheme and the scheme proposed by Zhang *et al.* (Zhang et al., 2019).

Finding an optimal solution to K-Means is NP-Hard (Mahajan, Nimbhorkar, & Varadarajan, 2009), therefore, many similar but alternative algorithms have been proposed, the most



(a) FPA_k Classifier

(b) $d^* - privacy$ Classifier

(c) *NDA* Classifier

(d) Legends

Figure 6.4: Accuracy Across Multiple Epochs For Three Classifiers

commonly used (and used in this paper) is Lloyd's Algorithm. The time complexity of Lloyd's algorithm is $O(t \times k \times n \times d)$ (Hartigan & Wong, 1979) where t is a number of iterations over n points in d dimensions with k number of clusters. The time complexity of our algorithm is $O(k \times k_z \times v)$, where k is the number of clusters. For each cluster, defined as $\mathbf{C}_j$, the value $|\mathbf{C}_j|$ must be evaluated, which must be done iteratively. k_z is the number of clusters where $|\mathbf{C}_j| > 1$ so that the potential waste and deficit of the video V will be evaluated relative to each centroid. The value v represents the time taken to find the maximum and minimum values of a data vector so that the difference between $\mathbf{x}$ and $\mathbf{y}$ can be calculated

Table 6.2: Execution Time (Millisecond) and Complexity

Scheme	Time (ms)	Time Complexity
NDA	16.1	$O(k \times k_z \times v)$
NDA w/o Pre-fit Algorithm	136.3	$O(t \times 2k \times n \times d \times k_z \times v)$
FPA_k	8.4	N/A
FPA_k w/o Pre-Computation	616000	$O(n^2)$
$d^* - privacy$	155.1	$O(\lambda \times n)$

for waste and deficit. Therefore, the time complexity of *NDA* is $O(t \times 2k \times n \times d \times k_z \times v)$.

The lower bound of the time complexity of a Fast Fourier Transformation hasn't been proven, but through experimentation, we determined that the slowest component of the computation of FPA_k privacy is the calculation of sensitivity, which is defined as the greatest difference between any two data vectors in a data-set. Considering all videos $\mathbf{x} \in \mathbf{S}$ one must find the difference between all videos in a set relative to each other. For example, the difference between the first video compared to each other video, etc., so the time complexity of this calculation is $O(n^2)$. This is an important constraint, because the provable privacy of differential privacy is contingent on the value for sensitivity (Rastogi & Nath, 2010), so this value cannot be chosen randomly to speed up computation.

The time complexity of $d^* - privacy$ would be $O(\lambda \times n)$, because it performs a series of constant time computations for the length of one data vector, additionally, during each iteration it must compute $D(i)$. The time consuming component for our implementation was the calculation of $D(i)$ which is defined as the largest power of two that divides a number i (Xiao et al., 2015). We found this value iteratively, trying m numbers until we found the largest square that divided i, this made out implementation $O(n \times m)$, there are more efficient ways to find that value so it is defined as λ. The implementation of $d^* - privacy$ was considerably slower than both *NDA* and FPA_k pre-fit algorithms. Both FPA_k and *NDA* have an impressive computational performance when pre-computation of a component of each is considered. In the case of *NDA*, pre-fitting the K-Means algorithm considerably increased performance, in the case of FPA_k, pre-computation of the sensitivity increases performance considerably. Considering the time complexity of each solution, without pre-computation, FPA_k is not viable, and while $d^* - privacy$ is potentially viable, the waste added by this scheme can be a problem. Our algorithm will scale better compared to FPA_k if pre-computation isn't possible, and doesn't require a list of all videos for pre-computation, without a full list, $FPAK_k$ can't be proven to be differentially private. Results obtained through experimentation for the computation of $\mathbf{y}_i$ from $\mathbf{x}_i$ both with and without

pre-computation can be seen in Table 6.2. We recorded the time it took each method to obfuscated a single video, measuring both pre-computation times and non pre-computed times for FPA_k and NDA.

All the experiments were implemented using Python[1] version 3.7 and TensorFlow[2] version 2.3. Also, they were executed on a desktop equipped with Intel Core i7-6700 processor at 3.40GHz, 16GB memory, AMD Radeon(TM) R5 340x display adaptor, and Windows 10 Pro 64-bit Operating System.

[1] www.python.org
[2] www.tensorflow.org

Chapter 7

Proposed Scheme 2: A Single Cluster

7.1 Motivation

In order to create a robust scheme, we decided to create a client-side implementation in addition to the server side implementation. We initially used a server-side implementation in order to give the computational burden of our scheme to the server, since video streaming can be a computationally expensive task and we didn't want to add any additional burden to the client. Therefore, in order to implement a client-side solution, we had to consider a more computationally efficient scheme. Additionally, the client-side scheme needed to be able to be implemented in real time. If a client clicks on a video, the full pattern hasn't been streamed yet so it cannot be clustered. The server-side implementation can simply keep a database of video patterns in order to implement real time, but the client doesn't have this amount of storage. Therefore, our goal was to implement a computationally efficient scheme that can be implemented in real time.

7.2 Proposed Scheme Overview

In this scheme, the client will request video patterns from server, for example, 40 video patterns. The client has the flexibility to request more or less. The average of all these patterns

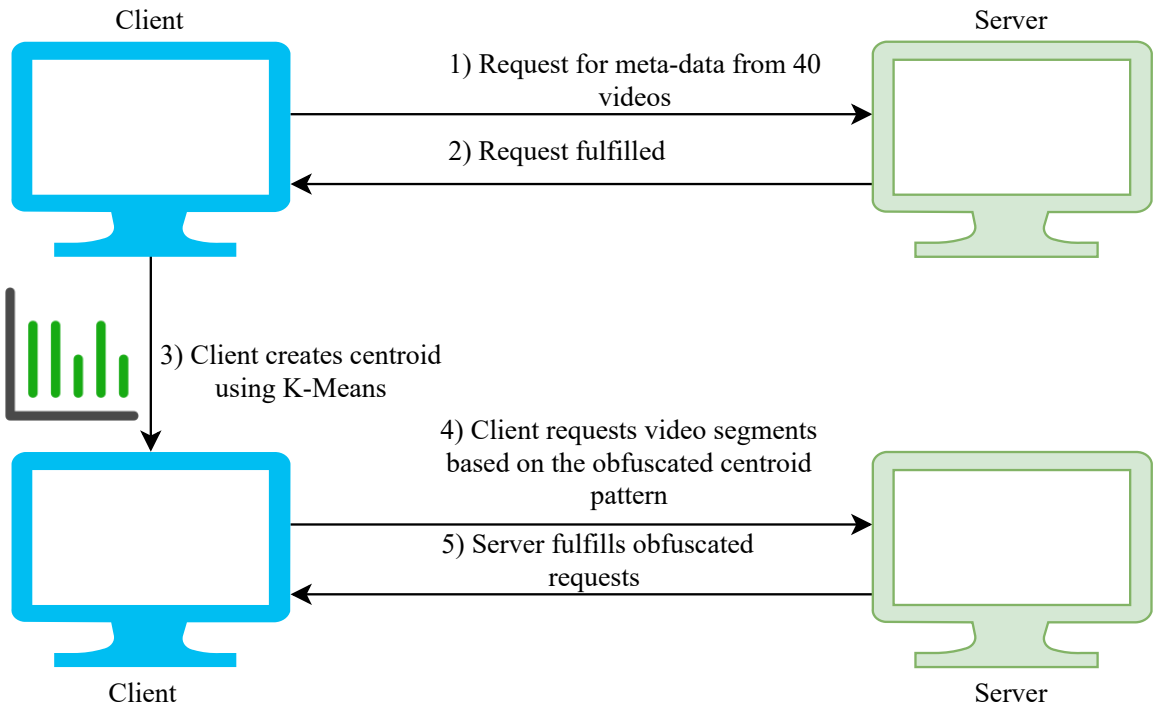


Figure 7.1: Proposed Scheme 2

will then be computed, which is equivalent to making a single cluster in K-Means clustering, therefore our privacy definition applies directly to this scheme. All videos watched by the client will then be obfuscated with this average. The client is able to request more videos than will be used for obfuscation, i.e., the client is able to request 100 videos, and create an average from only 50 of them. This way, the client is able to hide the values in the average from the server. An overview of our proposed scheme 2 can be seen in Figure 7.1.

7.3 Experimentation

In order to show the effectiveness of our Proposed Scheme 2, we performed two experiments. First, we performed K-Means clustering on our two different data-sets. One K-Means model was clustered on the 1000 video data-set, and one K-Means cluster was fit on the 41 video data-set. We performed clustering on these videos with multiple cluster sizes. For cluster sizes over 1, we performed obfuscation with NDA. We then measured the

accuracy of videos obfuscated with each of the two models at various cluster sizes. Results can be seen in Table 7.1 and Table 7.2.

We also performed obfuscation of videos in order to show that our Proposed Scheme 2 meets our privacy definition. For this experiment we obfuscated a set number of videos using Proposed Scheme 2, then tested the accuracy of the CNN against these videos. For example, we used 10 videos for our Proposed Scheme 2 obfuscation pattern, then tested the CNN accuracy against only these 10 videos after being obfuscated. According to our privacy definition, the number of videos used for obfuscation should define the accuracy of an adversary against our proposed scheme. If 10 videos are obfuscated by a cluster with only 10 videos, the accuracy against our proposed scheme should be $\frac{1}{10}$ or 10%. Results from this experiment can be seen in Table 7.3.

7.4 Experimental Results

In Table 7.1 it can be seen that the accuracy of the scheme reduces when the cluster size increases past 2 clusters. The accuracy after the cluster size of 2 does not hold to our privacy definition. The same result can be seen in Table 7.2 showing that this trend holds across data-sets of different sizes.

Table 7.3 represents the clustering of our Proposed Scheme 2. It can be seen in this table that the accuracy rigorously conform to our privacy definition for every cluster size. Additionally, the waste and deficit are slightly lower for our Proposed Scheme 2 than for *NDA*.

Table 7.1: Accuracy Across Multiple Cluster Sizes 1000 Video Data-set *NDA*

Number of Clusters	**Accuracy**	**Waste + Deficit**
1	0.0243	2.278
2	0.0243	2.284
3	0.0414	2.297
4	0.0414	2.298
5	0.0414	2.297
6	0.0414	2.295

Table 7.2: Accuracy Across Multiple Cluster Sizes 41 Video Data-set *NDA*

Number of Clusters	**Accuracy**	**Waste + Deficit**
1	0.0243	2.324
2	0.0487	2.329
3	0.0463	2.330
4	0.0487	2.318
5	0.0463	2.317
6	0.0463	2.339

Table 7.3: Accuracy Across Multiple Cluster Sizes 41 Video Data-set Proposed Scheme 2

Cluster Size	**Accuracy**	**Waste + Deficit**
2	0.5	1.704
3	0.33	2.162
4	0.25	2..194
10	0.10	2.373
20	0.05	2.523
40	0.025	2.111

Chapter 8

Related Works

8.1 MPEG-DASH Leak

There have been multiple traffic analysis attacks that exploit the MPEG-DASH leak on video streaming. The most effective attack with the broadest attack surface is by Schuster *et al.* (Schuster et al., 2017). The attack can be implemented in the form of a malicious web advertisement written in JavaScript. Additionally, no closed world assumptions were imposed on the attack model, the adversary can identify the target video without need for a predetermined ”set” of videos. The authors' train a Convolutional Neural Network (CNN) for target video identification and achieve high accuracy and precision, with the accuracy of the accuracy of their YouTube identifier at 99.4% depending on the features selected for training.

Implementing algorithmic approaches has also been effective at video fingerprint for identification, Gu *et al.* (Gu et al., 2019) achieved up to 90% accuracy using a variant of Dynamic Time Warping. Dynamic Time Warping is an algorithm for comparing time series data, this algorithm was implemented to determine the similarity between a known video traffic pattern and an unknown video fingerprint to determine the identity of the unknown video from a set of possible candidates.

Instead of dynamic time warping to determine similarity, Reed *et al.* (Reed & Klimkowski, 2016) used a multi stage algorithm that breaks videos into candidate ”windows” that have a similar throughput to the target video. After selecting potential candidates, Pearson's correlation coefficient is used to determine a ”match” between two video fingerprints. The model achieved an accuracy of 96%.

Finally, Dubin *et al.* (Dubin et al., 2017) also used machine learning to great effect, employing nearest neighbor, nearest neighbor to class algorithm, and support vector machines and achieved an accuracy above 95%.

In response to these attacks, specifically the CNN based attack, Zhang *et al.* (Zhang et al., 2019) used differential privacy as a defense mechanism to obfuscate video bit-rate data.

8.2 Traffic Analysis

A traffic analysis attack is a form of attack in which an adversary learns information by spying on a victims network traffic. Traffic analysis attacks have a broad set of goals. Some seek to compromise information about a victims smart home for theft or other malicious purposes (Copos, Levitt, Bishop, & Rowe, 2016; Kennedy, 2019). Some seek to compromise privacy of a victim unsuspecting victim (Taylor, Spolaor, Conti, & Martinovic, 2018; Meidan et al., 2017; Skowron, Janicki, & Mazurczyk, 2020; Li et al., 2016; Feghhi & Leith, 2016). Some traffic analysis attacks target victims that are using an anonymous browsing service such as Tor or Pishon (Yang, Gu, Ling, Yin, & Luo, 2017; Ejeta & Kim, 2017; Basyoni, Fetais, Erbad, Mohamed, & Guizani, 2020; Attarian, Abdi, & Hashemi, 2019; Abe & Goto, 2016). Some of these attacks are even able to determine the IP address of users on Tor (Iacovazzi, Frassinelli, & Elovici, 2019).

Since most web traffic is encrypted these days, most traffic analysis attacks often rely on side channel information and machine learning to be effective. Some side channel reliant traffic analysis attacks use only on timing information (Feghhi & Leith, 2016; Ramesh

& Prakash, 2017; Monaco, 2019) while others depend on different side channel information, such as presence or absence of communication information (Baroutis & Younis, 2016), delaying and analyzing HTTP requests (Monaco, 2019), and standard side channel information, such as packet length, number of packets, time, etc. (Abe & Goto, 2016). Machine learning allows adversaries to analyze even encrypted traffic to steal user information (Msadek, Soua, & Engel, 2019; Kennedy, 2019; Taylor, Spolaor, Conti, & Martinovic, 2017; Kausar, Aljumah, Alzaydi, & Alroba, 2019).

Defending against traffic analysis attacks can be difficult, as noted previously, encryption is not enough. Some work done seeks to make an efficient defense using Adaptive Padding (Juarez, Imani, Perry, Diaz, & Wright, 2016). To defend against website fingerprinting, some researchers (Wang & Goldberg, 2017) modify the way browsers communicate, allowing burst sequences to be molded more easily. Privacy can be added at the network layer by adding latency (Chen et al., 2018), controlling the network latency to allow for privacy and utility. Differential privacy also presents a viable solution for obfuscating traffic from an adversary. Differential privacy can also be employed (Liu, Zhang, & Fang, 2018) to protect smart homes from traffic analysis attacks.

Chapter 9

Conclusion

This paper aimed to develop a privacy preservation scheme that conformed to rigorous privacy standards while having a high computational efficiency, overcoming the common trade-off between privacy and computational speed. Using K-Means clustering as a base, we created our own algorithm, named No Data Are Alone, that accomplished this goal. Our algorithm provided privacy at a higher level when measures against the most robust attack method, which relied on a Convolutional Neural Network (CNN). We created multiple CNNs, each trained on data obfuscated by a different scheme. The attack CNN being trained on data obfuscated by our scheme never improved in accuracy and was ever able to reach an accuracy above 7.07%. Other differential privacy defense techniques were vulnerable to a CNN trained against them, and the CNNs trained against these schemes had 20% or greater increases in accuracy. Additionally, the computational cost, measured in $waste + deficit$, of our scheme was less than half of the best performing scheme.

References

Abe, K., & Goto, S. (2016). Fingerprinting attack on tor anonymity using deep learning. *Proceedings of the Asia-Pacific Advanced Network*, *42*, 15–20.

Arthur, D., & Vassilvitskii, S. (2006). *k-means++: The advantages of careful seeding* (Tech. Rep.). Stanford.

Attarian, R., Abdi, L., & Hashemi, S. (2019). Adawfpa: Adaptive online website fingerprinting attack for tor anonymous network: A stream-wise paradigm. *Computer Communications*, *148*, 74–85.

Baroutis, N., & Younis, M. (2016). A novel traffic analysis attack model and base-station anonymity metrics for wireless sensor networks. *Security and Communication Networks*, *9*(18), 5892–5907.

Basyoni, L., Fetais, N., Erbad, A., Mohamed, A., & Guizani, M. (2020). Traffic analysis attacks on tor: A survey. In *2020 ieee international conference on informatics, iot, and enabling technologies (iciot)* (p. 183-188).

Chan, T.-H. H., Shi, E., & Song, D. (2011). Private and continual release of statistics. *ACM Transactions on Information and System Security (TISSEC)*, *14*(3), 1–24.

Chen, C., Asoni, D. E., Perrig, A., Barrera, D., Danezis, G., & Troncoso, C. (2018). Taranet: Traffic-analysis resistant anonymity at the network layer. In *2018 ieee european symposium on security and privacy (euros p)* (p. 137-152).

Copos, B., Levitt, K., Bishop, M., & Rowe, J. (2016). Is anybody home? inferring activity from smart home network traffic. In *2016 ieee security and privacy workshops (spw)*

(p. 245-251).

Dubin, R., Dvir, A., Pele, O., & Hadar, O. (2017). I know what you saw last minute—encrypted http adaptive video streaming title classification. *IEEE Transactions on Information Forensics and Security*, *12*(12), 3039-3049.

Dwork, C., McSherry, F., Nissim, K., & Smith, A. (2006). Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference* (pp. 265–284).

Ejeta, T. G., & Kim, H. J. (2017). Website fingerprinting attack on psiphon and its forensic analysis. In *International workshop on digital watermarking* (pp. 42–51).

Feghhi, S., & Leith, D. J. (2016). A web traffic analysis attack using only timing information. *IEEE Transactions on Information Forensics and Security*, *11*(8), 1747-1759.

Gu, J., Wang, J., Yu, Z., & Shen, K. (2019). Traffic-based side-channel attack in video streaming. *IEEE/ACM Transactions on Networking*, *27*(3), 972-985.

Hartigan, J. A., & Wong, M. A. (1979). Algorithm as 136: A k-means clustering algorithm. *Journal of the royal statistical society. series c (applied statistics)*, *28*(1), 100–108.

Iacovazzi, A., Frassinelli, D., & Elovici, Y. (2019, September). The DUSTER attack: Tor onion service attribution based on flow watermarking with track hiding. In *22nd international symposium on research in attacks, intrusions and defenses (RAID 2019)* (pp. 213–225). Chaoyang District, Beijing: USENIX Association. Retrieved from `https://www.usenix.org/conference/raid2019/presentation/iacovazzi`

Juarez, M., Imani, M., Perry, M., Diaz, C., & Wright, M. (2016). Toward an efficient website fingerprinting defense. In *European symposium on research in computer security* (pp. 27–46).

Kadloor, S., Kiyavash, N., & Venkitasubramaniam, P. (2016). Mitigating timing side channel in shared schedulers. *IEEE/ACM Transactions on Networking*, *24*(3), 1562-1573.

Kausar, F., Aljumah, S., Alzaydi, S., & Alroba, R. (2019). Traffic analysis attack for identifying users' online activities. *IT Professional*, *21*(2), 50-57.

Kennedy, S. M. (2019). *Encrypted traffic analysis on smart speakers with deep learning* (Unpublished doctoral dissertation). University of Cincinnati.

Li, H., Xu, Z., Zhu, H., Ma, D., Li, S., & Xing, K. (2016). Demographics inference through wi-fi network traffic analysis. In *Ieee infocom 2016 - the 35th annual ieee international conference on computer communications* (p. 1-9).

Liu, J., Zhang, C., & Fang, Y. (2018). Epic: A differential privacy framework to defend smart homes against internet traffic analysis. *IEEE Internet of Things Journal*, *5*(2), 1206–1217.

Lloyd, S. (1982). Least squares quantization in pcm. *IEEE transactions on information theory*, *28*(2), 129–137.

MacQueen, J., et al. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth berkeley symposium on mathematical statistics and probability* (Vol. 1, pp. 281–297).

Mahajan, M., Nimbhorkar, P., & Varadarajan, K. (2009). The planar k-means problem is np-hard. In *International workshop on algorithms and computation* (pp. 274–285).

Meidan, Y., Bohadana, M., Shabtai, A., Guarnizo, J. D., Ochoa, M., Tippenhauer, N. O., & Elovici, Y. (2017). Profiliot: A machine learning approach for iot device identification based on network traffic analysis. In *Proceedings of the symposium on applied computing* (p. 506–509). New York, NY, USA: Association for Computing Machinery. Retrieved from `https://doi.org/10.1145/3019612.3019878` doi: 10.1145/3019612.3019878

Monaco, J. V. (2019). Feasibility of a keystroke timing attack on search engines with autocomplete. In *2019 ieee security and privacy workshops (spw)* (p. 212-217).

Msadek, N., Soua, R., & Engel, T. (2019). Iot device fingerprinting: Machine learning based encrypted traffic analysis. In *2019 ieee wireless communications and networking conference (wcnc)* (p. 1-8).

Ramesh, S., & Prakash, D. S. A. (2017). An effective attack analysis and defense in web

traffic using only timing information..

Rao, A., Legout, A., Lim, Y.-s., Towsley, D., Barakat, C., & Dabbous, W. (2011). Network characteristics of video streaming traffic. In *Proceedings of the seventh conference on emerging networking experiments and technologies* (pp. 1–12).

Rastogi, V., & Nath, S. (2010). Differentially private aggregation of distributed time-series with transformation and encryption. In *Proceedings of the 2010 acm sigmod international conference on management of data* (pp. 735–746).

Reed, A., & Klimkowski, B. (2016). Leaky streams: Identifying variable bitrate dash videos streamed over encrypted 802.11n connections. In *2016 13th ieee annual consumer communications networking conference (ccnc)* (p. 1107-1112).

Schuster, R., Shmatikov, V., & Tromer, E. (2017). Beauty and the burst: Remote identification of encrypted video streams. In *26th {USENIX} security symposium ({USENIX} security 17)* (pp. 1357–1374).

Skowron, M., Janicki, A., & Mazurczyk, W. (2020). Traffic fingerprinting attacks on internet of things using machine learning. *IEEE Access*, *8*, 20386-20400.

Sodagar, I. (2011). The mpeg-dash standard for multimedia streaming over the internet. *IEEE MultiMedia*, *18*(4), 62-67.

Steinhaus, H. (1956). Sur la division des corps matériels en parties. *Bull. Acad. Polon. Sci*, *1*(804), 801.

Taylor, V. F., Spolaor, R., Conti, M., & Martinovic, I. (2017). Robust smartphone app identification via encrypted network traffic analysis. *IEEE Transactions on Information Forensics and Security*, *13*(1), 63–78.

Taylor, V. F., Spolaor, R., Conti, M., & Martinovic, I. (2018). Robust smartphone app identification via encrypted network traffic analysis. *IEEE Transactions on Information Forensics and Security*, *13*(1), 63-78.

Wang, T., & Goldberg, I. (2017, August). Walkie-talkie: An efficient defense against passive website fingerprinting attacks. In *26th USENIX security symposium (USENIX*

security 17) (pp. 1375–1390). Vancouver, BC: USENIX Association. Retrieved from https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presenta

Xiao, Q., Reiter, M. K., & Zhang, Y. (2015). Mitigating storage side channels using statistical privacy mechanisms. In *Proceedings of the 22nd acm sigsac conference on computer and communications security* (pp. 1582–1594).

Yang, M., Gu, X., Ling, Z., Yin, C., & Luo, J. (2017). An active de-anonymizing attack against tor web traffic. *Tsinghua Science and Technology*, *22*(6), 702-713.

Zhang, X., Hamm, J., Reiter, M. K., & Zhang, Y. (2019). Statistical privacy for streaming traffic. In *Ndss*.