Hindawi
Security and Communication Networks
Volume 2021, Article ID 4702469, 14 pages
https://doi.org/10.1155/2021/4702469

Research Article

Efficient yet Robust Privacy Preservation for MPEG-DASH-Based Video Streaming

Luke Cranfill,[1] **Jeehyeong Kim**,[2] **Hongkyu Lee**,[1] **Victor Youdom Kemmoe**,[1] **Sunghyun Cho**,[3] **and Junggab Son**[1]

[1]*Information and Intelligent Security Lab, Kennesaw State University, Marietta, GA 30060, USA*
[2]*Korea Electronics Technology Institute, Seongnam-si, Gyeonggi-do 13509, Republic of Korea*
[3]*Department of Computer Science and Engineering, Hanyang University, Ansan, Gyeonggi-do 15588, Republic of Korea*

Correspondence should be addressed to Junggab Son; json@kennesaw.edu

Received 18 May 2021; Accepted 12 August 2021; Published 27 August 2021

Academic Editor: Feng Xue

MPEG-DASH is a video streaming standard that outlines protocols for sending audio and video content from a server to a client over HTTP. 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 of data and the time the server sent the data to the user. After a fingerprint of this data is created, the adversary can use this to identify whether a target user is watching the corresponding video. Only one defense strategy has been proposed to deal with this problem: differential privacy that adds sufficient noise in order to muddle the attacks. However, that strategy still suffers from the trade-off between privacy and efficiency. This paper proposes a novel defense strategy against the attacks with rigorous privacy and performance goals creating a private, scalable solution. Our algorithm, "No Data are Alone" (NDA), is highly efficient. The experimental results show that our scheme is more than two times efficient in terms of excess downloaded video (represented as waste) compared to the most efficient differential privacy-based scheme. Additionally, no classifier can achieve an accuracy above 7.07% against videos obfuscated with our scheme.

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 and Netflix, 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 both for recreation and as an educational hub, there are many things a user might not want to expose. 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 that 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 will require a lower bit-rate. MPEG-

DASH breaks videos into time segments of approximately the same length [1], 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 [2–5].

The most effective of these attacks was introduced by Schuster et al. [2]; its effectiveness is due to the fact that it makes no closed world assumptions, has high accuracy, and 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 is used against schedulers [6]. This saturation allows the adversary to learn the victim 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 [2].

The defense against 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. [7]. This paper focused on defending against the CNN attack model mentioned above [2]. The work done in this paper uses differential privacy, specifically d^* – privacy [8], which is adjusted for time series data, and the Fourier Perturbation Algorithm (FPA_k) which was proposed by Rastogi and Nath [9]. 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, 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, so this pattern would be representative of many videos and thus 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 defense scheme "No Data are Alone" (NDA), an augmented version of K-Means, clusters 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 formally. We recreate the CNN video classifier [2] to show our scheme's effectiveness. In addition to the experimentation done to show our scheme's privacy, we give a formal privacy definition that is based on the L_1 norm, in order to give a broader view of privacy beyond just one attack. When compared to differential privacy, our method significantly reduces computational waste while providing higher privacy.

The contributions of our paper are summarized as follows:

(i) An effective and novel defense scheme that generates an optimal request pattern called "No Data are Alone" (NDA) is proposed

(ii) The time complexity of our scheme is compared with the differential privacy schemes; it has not been shown previously for the differentially private schemes

(iii) Multiple attack CNNs are created and trained on unobfuscated data and noised data, and detailed explanations of how they are trained are given.

(iv) These CNNs are then used for a thorough evaluation f privacy provided by our scheme compared to differentially private schemes

(v) The privacy of both schemes is evaluated using a privacy definition based on the L_1 norm, and the results show that our scheme outperforms differential privacy

The remainder of this paper is organized as follows: Section 2 gives necessary background knowledge, and then Section 3 lays out the problem description. Section 4 introduces our proposed scheme in detail, and Section 5 details implementation and experimental methods. Section 6 shows the results of our evaluations. Section 7 introduces some notable related research results. Finally, we conclude this paper in Section 8.

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 [1].

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, with 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 of variable sizes or held constant [1] 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 [10] because there are periods when new segments are being requested, which causes 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 uses this data as a fingerprint. A visualization of a video's unobfuscated bit-rate over time can be seen in Figure 1.

2.2. K-Means. 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 $\mathscr{C} = \{\mathbf{C}_1, \mathbf{C}_2, \ldots, \mathbf{C}_k\}$. Once $\mathscr{C}$ 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 $\mathbf{C}_i \in \mathscr{C}$, $\mathbf{m}_i \in \mathbf{M}$ is considered to be its centroid. After the initialization of means, each vector $\mathbf{x}_i \in \mathbf{S}$ is assigned to a cluster $\mathbf{C}_j \in \mathscr{C}$ 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 (1) is computed. Then, $\mathbf{x}_i$ is assigned to the cluster $\mathbf{C}_j$ whose centroid $\mathbf{m}_j$ is the closest to $\mathbf{x}_i$.

$$\operatorname{dist}\left(\mathbf{x}_i, \mathbf{m}_j\right) = \left\|\mathbf{x}_i - \mathbf{m}_j\right\|^2. \tag{1}$$

After the assignment phase, each cluster $\mathbf{C}_i \in \mathscr{C}$ 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 as in the following equation:

$$\mathbf{m}_j = \frac{1}{\left|\mathbf{C}_j\right|} \sum_{i=1}^{\left|\mathbf{C}_j\right|} \mathbf{x}_i^{\left(\mathbf{C}_j\right)}, \tag{2}$$

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 remains constant.

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. During 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 [2]. 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, both algorithmic and machine learning based, have been taken to use this data for malicious purposes. The algorithmic approaches [3, 4] seek to measure similarity between the user's bit-rate data and the adversary's prerecorded bit-rate data for a specific video. The machine learning approaches [2, 5] 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 [2]. More information about the attacks and their implementation is included in Section 7.

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 unobfuscated graph in Figure 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 [2], 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. [7]. 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. [7] leverages two

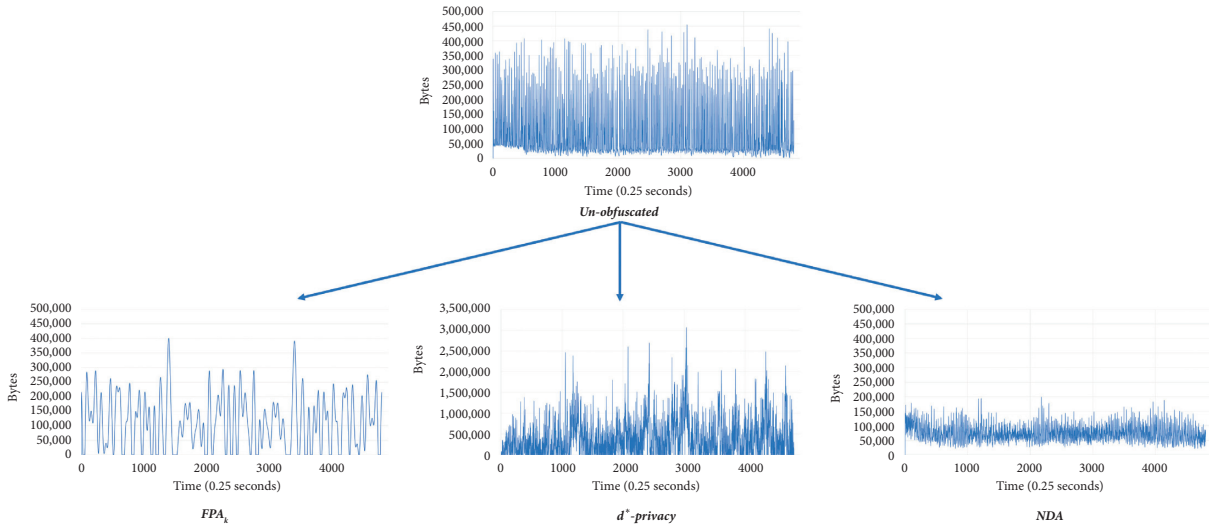

Figure 1: Obfuscation pattern comparison.

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. [2], 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. We sought to create a formal privacy definition to reach beyond the privacy shown through experimental results and give a more in depth view of the privacy being provided. Drawing on the attack paper by Schuester et al. [2], 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:

$$L_1(\mathbf{a}, \mathbf{b}) = \|\mathbf{a} - \mathbf{b}\|_1 = \sum_{i=1}^{n} |\mathbf{a}[i] - \mathbf{b}[i]|. \quad (3)$$

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. [2], the adversary was successfully able to identify a video if the L_1 norm between the recorded data v 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 [7]. For our own scheme, privacy is twofold. 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 our scheme, 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, we show our scheme's privacy 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.

4. Proposed Scheme

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. [7], 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 and that does not need to be. Let $\mathbf{a} \in \mathbb{R}^n$ be a vector that represents the original unobfuscated video pattern and let $\mathbf{b} \in \mathbb{R}^n$ be a vector that represents the obfuscated video pattern.

$$\begin{aligned} \text{waste} &= \max_{1 \le i \le n} \{\max(\mathbf{b}[i] - \mathbf{a}[i], 0)\}, \\ \text{deficit} &= \max_{1 \le i \le n} \{\max(\mathbf{a}[i] - \mathbf{b}[i], 0)\}. \end{aligned} \tag{4}$$

4.2. 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 2 depicts our overall scheme in detail, and Table 1 gives definitions of notations used in this section to define our scheme. 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 video i sent from the server to the client. Let $\mathbf{S} = (\mathbf{x}_1, \ldots, \mathbf{x}_l)$ 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 whose 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 $\mathbf{x}_{i,n}$ 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 $\mathbf{x}_i$ (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.

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. Accordingly, 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 $\mathbf{m}_j = (m_{j,1}, m_{j,2}, \ldots, m_{j,n})$ that has identical dimensions with video segments and is calculated by (2), 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 3).

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

$$\begin{aligned} \mathbf{m} &= \frac{1}{|\mathbf{C}|} \sum_{i=1}^{|\mathbf{C}|} \mathbf{x}_i^{(\mathbf{C})} \\ &= \mathbf{x}, \end{aligned} \tag{5}$$

where $|\mathbf{C}| = 1$ and $\mathbf{x}_i^{(\mathbf{C})}$ indicates $\mathbf{x}_i \in \mathbf{C}$.

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 $(\mathbf{x}_{i,1}, \mathbf{x}_{i,2}, \ldots, \mathbf{x}_{i,n}) \in \mathbf{x}_i$ for some video $\mathbf{x}_i$, and the server will fulfill these requests with an obfuscated pattern $\mathbf{y}_i$. 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 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 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 patterns $\mathbf{M} = \{\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_k\}$. Our algorithm can be represented as follows:

$$\mathbf{C}_j := \operatorname{argmin}\left(\max_{1 \le i \le n} \left\{ \max_{j \in k} \left(\mathbf{m}_{j,n} - \mathbf{x}_{i,n}, 0 \right) \right\} + \max_{1 \le i \le n} \left\{ \max_{j \in k} \left(\mathbf{x}_{i,n} - \mathbf{m}_{j,n}, 0 \right) \right\} \right). \tag{6}$$

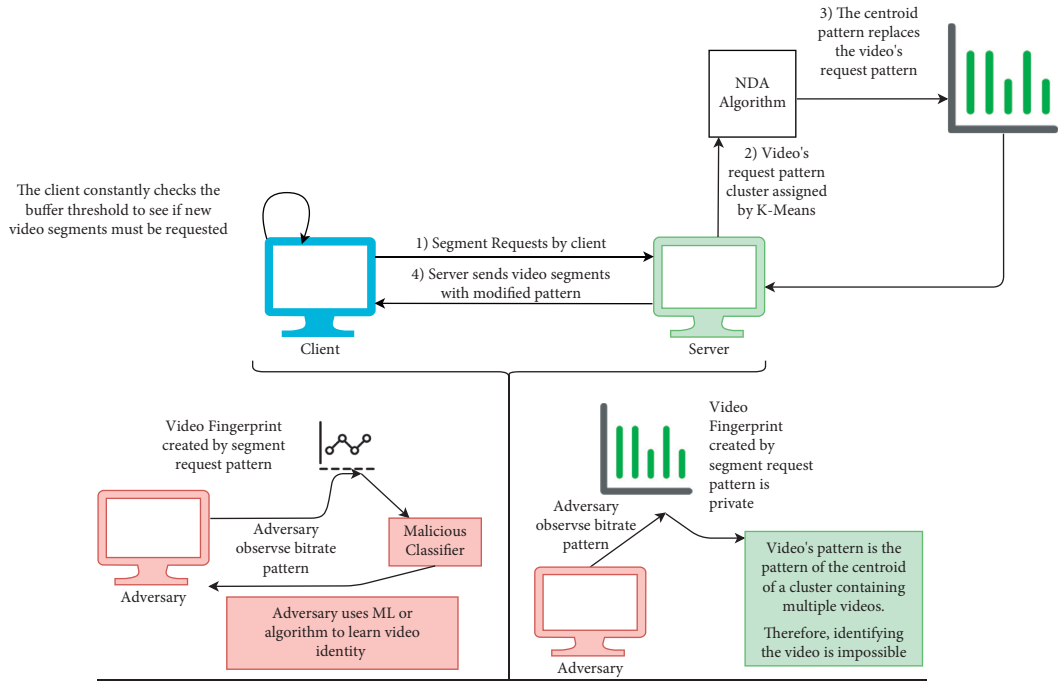

Figure 2: A conceptual overview of the proposed scheme.

Table 1: Notations.

Notation	Description
$\mathbf{x}_i$	i-th video capture
$\mathbf{x}_{i,n}$	n-th segment of i-th video capture
$\mathbf{S}$	Set of all videos
N	K-Means model for NDA
$\mathbf{y}_i$	Single obfuscated video capture
$\mathbf{C}_j$	Cluster j
$\mathbf{m}_j$	Centroid of cluster j
$\mathbf{m}_{j,n}$	Centroid coordinate of n-th dimension
$\mathbf{M}$	Set of all cluster centroids

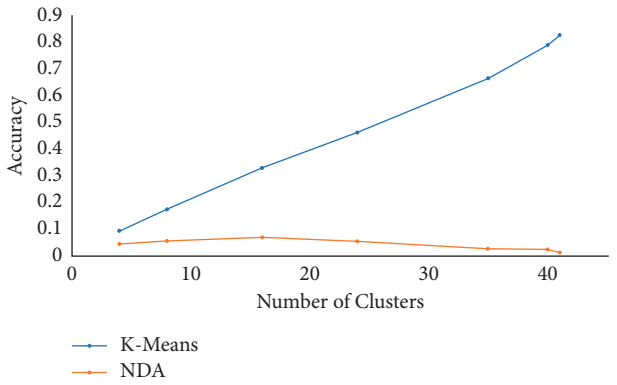

Figure 3: Accuracy of the CNN attack against K-Means and NDA across an increasing number of clusters.

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 $\mathbf{C}_j$. $\mathbf{x}$ represents 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.

4.3. Client Side vs. Server Side. In our scheme, the defense is implemented at server side. We chose server side implementation because we considered that the server already has full knowledge of the video the user is watching and any information about the user that has been given to the site either voluntarily (i.e., creating an account) or automatically (i.e., IP address). Additionally, the server side implementation is ideal from a computation perspective, giving the burden to the server and not the client. This is especially important considering the high number of users who access video streaming websites from mobile devices.

However, it can be argued that there are privacy advantages to considering the server as an untrustworthy third party. When considering this, we postulate that our scheme could be implemented on the client side. For our scheme to work on the client side, we must implement it using one of two ways: In the first way, the client side application must have knowledge from the server about the possible videos that are going to be watched, so that the client side can perform cluster prediction. In order for our scheme to be executed, there must be a prefit NDA model. This prefit model will be downloaded onto the client side with a browser extension that controls the implementation. Each prefit NDA model will have random initialization so that each model is unique. Over time, the model will refit as the client watches more videos, making it totally unique. When a client searches a video, the video provider shows a long list of videos that match search suggestions. The client would then query the video provider for the bit-rate pattern of these videos. The bit-rate pattern of each (starting with the best

Input: $\mathbf{M}$, $\mathbf{x}$
(1) let min_waste = 0
(2) let min_deficit = 0
(3) let $\mathbf{y} \in \mathbb{R}^n$
(4) for $\mathbf{m}_k$ in M where $k = (1, \ldots, l)$ do
let $\mathbf{C}$ be the cluster corresponding to $\mathbf{m}_k$
(5) if $|\mathbf{C}| = 1$ then
(6) remove $\mathbf{m}$ from $\mathbf{M}$
end
end
(7) for $\mathbf{m}_k$ in M where $k = (1, \ldots, l)$ do
(8) waste = $\max_{1 \le i \le n} \{\max(\mathbf{m}_{k,i} - \mathbf{x}_i, 0)\}$
(9) deficit = $\max_{1 \le i \le n} \{\max(\mathbf{x}_i - \mathbf{m}_{k,i}, 0)\}$
(10) if waste < min_waste and deficit < min_deficit then
(11) min_waste = waste
(12) min_deficit = deficit
(13) $\mathbf{y} = \mathbf{m}_k$
end
end
(14) return $\mathbf{y}$

Algorithm 1: No Data are Alone (NDA).

match) would be predicted by the NDA algorithm to determine the centroid. The video request pattern would then be changed to that of the centroid.

Alternatively, the client side application could cluster the video based on only the first 10 seconds of the video, not enough for the adversary to use for classification. The video would stream regularly for the first 10 seconds, and then this data would have its cluster predicted by the prefit NDA model on the client side. After successful prediction, the video request pattern would then be changed to the predicted cluster's centroid.

4.4. Cluster Recreation Probability. Our proposed scheme 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 on 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 by 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++ [11]. 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 and 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 that is 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 dataset. This means the adversary has at most a 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.

5. Implementation and Simulation

5.1. Data Collection. Data collection was automated using tshark by Wireshark (http://www.wireshark.org) and Selenium (http://www.selenium.dev). 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 dataset of 41 different hand selected videos. The bit-rate data of each of 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 dataset 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, with each being 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 were not used during experimentation but are used in Figure 4 to elaborate on our scheme. The collected dataset is available from our website (IIS Lab: http://i2s.kennesaw.edu/resources.html).

5.2. Comparison of Defense Schemes. To evaluate the performance of our scheme NDA compared to the proposed scheme of Zhang et al. [7], 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 as 10, and their video length was 720. Our video length was 4800, so accordingly we increased k to 67.

The unobfuscated graph in Figure 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, and 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 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 unobfuscated 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 unobfuscated 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 1.

The FPA_k graph in Figure 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 does not 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 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 unobfuscated 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 download of video data. This result agrees with the waste and deficit measurements given in Section 6.

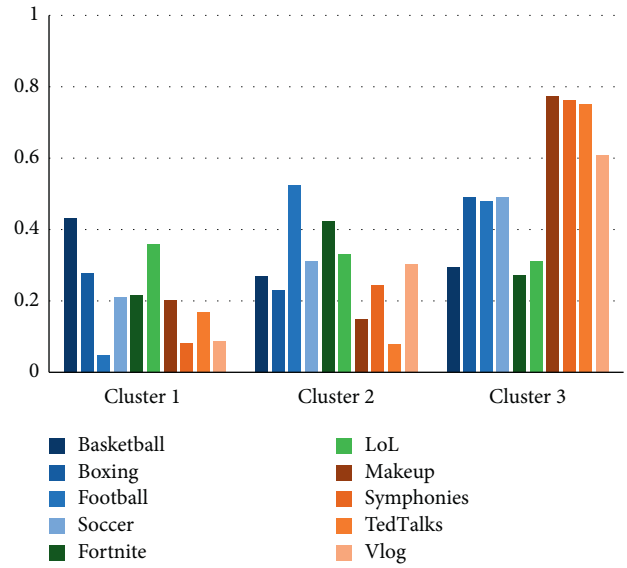

Figure 4: Video category distribution across multiple clusters.

5.3. Various Clustering Algorithms. While doing experimentation on K-Means clustering, we also performed clustering with two other popular clustering algorithms, Agglomerative and DB-SCAN clustering. When performing clustering, we noticed that both algorithms tended to generate the exact same pattern, such that they provided very lopsided clusters; most of the videos were grouped into a single cluster with only a few videos falling into other clusters. We tried to fit the models multiple times but the result was always the same. Since the output and the performance of all three clustering algorithms were nearly identical, 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.

5.4. Attack Classifiers. To test our proposed algorithm, we implemented the CNN created by Schuster et al. [2], 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 dataset 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 is a visual representation of these evaluation methods.

We created three "attack CNNs" and trained one on noised data from our scheme 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 2.

To train against our scheme, we used one dataset of videos obfuscated by our scheme and the original unobfuscated dataset. Instead of training a new classifier, we retrained the original model, Model_A. This new attack model, Model_B, was trained for 50 epochs; otherwise, significant overfitting occurred. This model theoretically should not 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 datasets. We included the original unobfuscated dataset and then 4 different datasets 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 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 of 0.0024 on the unobfuscated data.

To create the d^* − privacy attack model, we trained the Model_D from the original model Model_A with 2 datasets. We used the original unobfuscated data and one dataset obfuscated by d^* − privacy with an epsilon value of 0.0007. This model was trained for only 50 epochs; otherwise, overfitting 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 datasets; however, the accuracy of this model on the unobfuscated data was significantly lower than that of retraining the previous model and lower on data obfuscated by FPA_k that is was trained for classification. This implies that the knowledge about identifying unobfuscated data successfully transferred from one task (detecting unobfuscated 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.

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 dataset, both with our scheme, NDA, and with the two differential privacy schemes used by Zhang et al. [7]. We then represented the privacy level of each video with a Boolean value, either private or nonprivate based on the privacy definition given in Section 3.3. We then divided the total number of nonprivate videos for each scheme by the total number of videos, giving the percentage of nonprivate videos for each scheme. The results are shown in Table 3.

These results can also be interpreted inversely; that is, NDA with 4 clusters has a 1.4% chance of leaving a video nonprivate, or NDA has a 98.6% chance of successfully privatizing a video. It can also be seen that the privacy of our scheme degrades a little 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 2). When the epsilon value is increased slightly, the privacy degrades very quickly, more quickly than the attack accuracy using the CNN (see Table 2). 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 2).

6.2. Clusters and Video Types. Figure 4 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 that 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 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 percentage of videos in a given cluster; i.e., 40% of all Football videos fall into cluster 1, 20% of 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 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 dominates a single cluster.

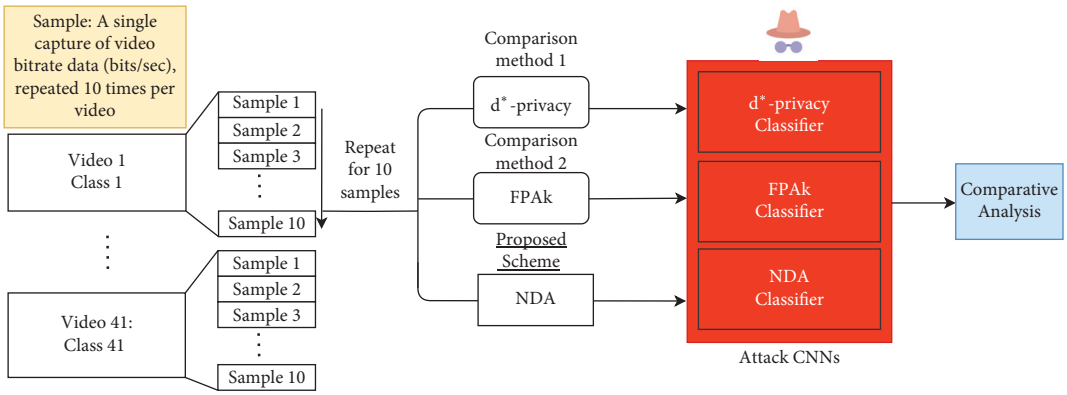

Figure 5: An overview of the evaluation methods.

Table 2: Comparative analysis: attack classifiers.

Accuracy range	FPA_k				$d^* - privacy$				NDA			
	ϵ	W	D	Acc	ϵ	W	D	Acc	C	W	D	Acc
0–10% accuracy	0.5	10.45	2.10	0.03	0.000005	29.51	1.68	0.03	4	0.15	2.17	0.07
10–20% accuracy	5	1.10	2.15	0.16	0.000014	10.43	1.42	0.18	N/A	N/A	N/A	N/A
21–40% accuracy	10	0.57	2.15	0.23	0.000018	8.13	1.31	0.33	N/A	N/A	N/A	N/A
40–60% accuracy	25	0.31	2.16	0.53	0.00002	7.31	1.25	0.44	N/A	N/A	N/A	N/A
> 60% accuracy	N/A	N/A	N/A	N/A	0.0001	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, and C: number of clusters.

Table 3: The percentage of nonprivate videos created by each scheme.

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

6.3. Traditional K-Means vs. NDA. Figure 3 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 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. The privacy of our proposed scheme, however, remains almost constant and the accuracy against our scheme never goes higher than 7%.

6.4. Accuracy vs. Waste + Deficit. Figure 6 is a comparison of the attack accuracy against each scheme vs. waste + deficit (measured in megabytes) of each scheme; each defensive scheme was tested against the classifier trained to attack it. For $d^* - privacy$ and FPA_k, we considered multiple epsilon values. Table 2 represents a summary of the performance of

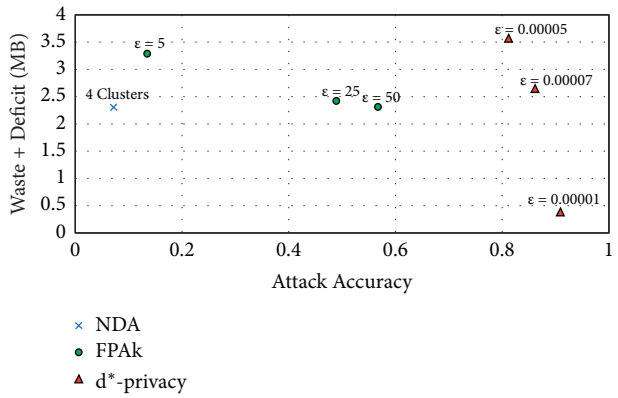

Figure 6: Accuracy vs. waste + deficit.

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.

In every epsilon case, our scheme outperforms differential privacy when considering both accuracy and waste + deficit. Our scheme significantly outperforms $d^* - privacy$ and creates 1 MB less waste + deficit while having an attack accuracy half as high as FPA_k. The correlation between epsilon, waste and deficit, and accuracy can also be clearly seen from this graph. Higher epsilon 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 have been equivalent to ours because the waste incurred was extremely high, and we wanted to preserve the scale of the graph (see Table 2).

6.5. Attack Accuracy across Epochs. Figure 7 shows the accuracy of the attack CNNs of each scheme across multiple epochs of training. In Figure 7, epoch 0 represents the classification accuracy of Model_A before training on noised data. Figure 7(c) depicts the accuracy of the classifier that was trained on both unobfuscated data and data that was privatized using our algorithm NDA. This result shows that over 50 epochs the classifier does not learn anything about the data and the accuracy remains below 8% for 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 7(a), 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 prediction accurately through the obfuscation. We used only 4 obfuscated datasets to produce this result, but it is logical to conclude that accuracy would grow higher against FPA_k if more obfuscated datasets were used. Additionally, we included one dataset of unobfuscated data while training the FPA_k attack classifier so that the model did not overfit and learn only FPA_k.

Figure 7(b) 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 scalability in the video streaming industry, we examined the computational overhead of both our scheme and the scheme proposed by Zhang et al. [7].

Finding an optimal solution to K-Means is NP-hard [12], so many similar but alternative algorithms have been proposed; the most commonly used (and used in this paper) is Lloyd's algorithm. The time complexity of Lloyd's algorithm is $O(t \times k \times n \times d)$ [13] 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 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 has not 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 dataset. 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 every 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 [9], 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 [8]. 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 prefit algorithms. Both FPA_k and NDA have an impressive computational performance when the precomputation of a component of each is considered. In the case of NDA, prefitting the K-Means algorithm considerably increased performance; in the case of FPA_k, precomputation of the sensitivity increases performance considerably. Considering the time complexity of each solution, without precomputation, 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 precomputation is not possible and does not require a list of all videos for precomputation; without a full list, FPAK_k cannot 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 precomputation can be seen in Table 4. We recorded the time it took each method to obfuscated a single video, measuring both precomputation times and nonprecomputed times for FPA_k and NDA.

All the experiments were implemented using Python (http://www.python.org) version 3.7 and TensorFlow (http://www.tensorflow.org) version 2.3. Furthermore, they were executed on a desktop equipped with Intel Core i7-6700 processor at 3.40 GHz, 16 GB memory, AMD Radeon™ R5 340x display adaptor, and Windows 10 Pro 64-bit operating system.

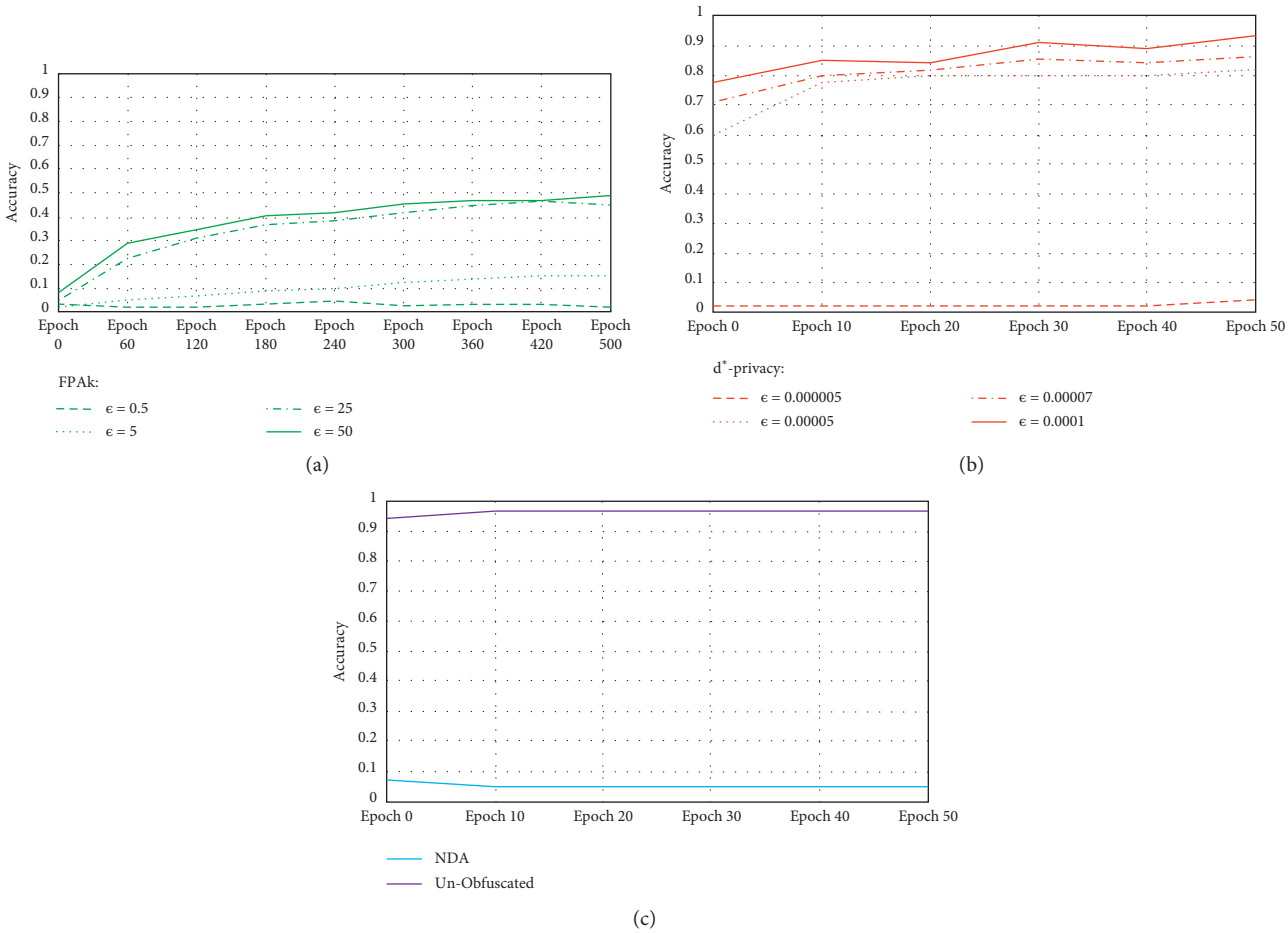

Figure 7: Accuracy across multiple epochs for three classifiers: (a) FPA_k classifier; (b) $d^* - privacy$ classifier; (c) NDA classifier.

Table 4: Execution time (millisecond) and complexity.

Scheme	Time (ms)	Time complexity
NDA	16.1	$O(k \times k_z \times v)$
NDA w/o prefit 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 Precomputation	616000	$O(n^2)$
d^* – privacy	155.1	$O(\lambda \times n)$

7. Related Works

7.1. 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 that proposed by Schuster et al. [2]. 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 their YouTube identifier being 99.4% depending on the features selected for training.

Implementing algorithmic approaches has also been effective in video fingerprinting for identification. Gu et al. [3] 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 using dynamic time warping to determine similarity, Reed and Klimkowski [4] used a multistage 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. [5] 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. [7] used differential privacy as a defense mechanism to obfuscate video bit-rate data.

7.2. Traffic Analysis. A traffic analysis attack is a form of attack in which an adversary learns information by spying on a victim's network traffic. Traffic analysis attacks have a broad set of goals. Some seek to compromise information about a victim's smart home for theft or other malicious purposes [14, 15]. Some seek to compromise privacy of an unsuspecting victim [16–20]. Some traffic analysis attacks target victims that are using an anonymous browsing service such as Tor or Psiphon [21–25]. Some of these attacks are even able to determine the IP address of users on Tor [26].

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 timing information [20, 27] while others depend on different side channel information, such as presence or absence of communication information [28], delaying and analyzing HTTP requests [27], and standard side channel information, such as packet length, number of packets, and time [25]. Machine learning allows adversaries to analyze even encrypted traffic to steal user information [15, 29–31].

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 [32]. To defend against website fingerprinting, some researchers [33] modify the way browsers communicate, allowing burst sequences to be molded more easily. Privacy can be added at the network layer by adding latency [34], 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 [35] to protect smart homes from traffic analysis attacks.

8. 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 measured 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 always 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 that of the best performing scheme.

Data Availability

The data used to support the findings of this study, which are bit-rate patterns of streaming videos, have been deposited in our lab repository (http://i2s.kennesaw.edu/resources.html). There are no restrictions, except commercial uses, placed on the usage of these data.

Disclosure

A part of this work was presented at the College of Computing and Software Engineering (CCSE) Computing Showcase (C-Day), Spring 2021 (Spring 2021 C-Day Program: https://ccse.kennesaw.edu/computing-showcase/cday-programs/spring2021program.php).

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the MSIT (Ministry of Science, ICT), Korea, under the High-Potential Individuals Global Training Program (2019-0-01601) supervised by the IITP (Institute for Information and Communications Technology Planning and Evaluation).

References

[1] I. Sodagar, "The mpeg-dash standard for multimedia streaming over the internet," *IEEE MultiMedia*, vol. 18, no. 4, pp. 62–67, 2011.

[2] R. Schuster, V. Shmatikov, and E. Tromer, "Beauty and the burst: Remote identification of encrypted video streams," in *Proceedings of the 26th USENIX Security Symposium (USENIXSecurity 17)*, pp. 1357–1374, Vancouver, Canada, August 2017.

[3] J. Gu, J. Wang, Z. Yu, and K. Shen, "Traffic-based side-channel attack in video streaming," *IEEE/ACM Transactions on Networking*, vol. 27, no. 3, pp. 972–985, 2019.

[4] A. Reed and B. Klimkowski, "Leaky streams: identifying variable bitrate dash videos streamed over encrypted 802.11n connections," in *Proceedings of the 2016 13th IEEE Annual Consumer Communications Networking Conference (CCNC)*, pp. 1107–1112, Las Vegas, NV, USA, September 2016.

[5] R. Dubin, A. Dvir, O. Pele, and O. Hadar, "I know what you saw last minute-encrypted HTTP adaptive video streaming title classification," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 12, pp. 3039–3049, 2017.

[6] S. Kadloor, N. Kiyavash, and P. Venkitasubramaniam, "Mitigating timing side channel in shared schedulers," *IEEE/ACM Transactions on Networking*, vol. 24, no. 3, pp. 1562–1573, 2016.

[7] X. Zhang, J. Hamm, M. K. Reiter, and Y. Zhang, "Statistical privacy for streaming traffic," in *Proceedings of the 26th*

Annual Network and Distributed System Security Symposium (NDSS 2019), San Diego, CA, USA, Feburary 2019.

[8] Q. Xiao, M. K. Reiter, and Y. Zhang, "Mitigating storage side channels using statistical privacy mechanisms," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pp. 1582–1594, Denver, CO, USA, October 2015.

[9] V. Rastogi and S. Nath, "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, Indianapolis, IN, USA, June 2010.

[10] A. Rao, L. Arnaud, Y. S. Lim, T. Don, C. Barakat, and W. Dabbous, "Network characteristics of video streaming traffic," in *Proceedings of the 7th Conference on emerging Networking Experiments and Technologies*, Tokyo, Japan, December 2011.

[11] D. Arthur and S. Vassilvitskii, "K-means++: the advantages of careful seeding," Technical report, Stanford University, Stanford, CA, USA, 2006.

[12] M. Mahajan, P. Nimbhorkar, and K. Varadarajan, "The planar k-means problem is np-hard, WALCOM: algorithms and computation," in *Proceedings of the International Workshop on Algorithms and Computation*, pp. 274–285, Springer, Kolkata, India, February 2009.

[13] J. A. Hartigan and M. A. Wong, "Algorithm as 136: a k-means clustering algorithm," *Applied Statistics*, vol. 28, no. 1, pp. 100–108, 1979.

[14] B. Copos, K. Levitt, M. Bishop, and J. Rowe, "Is anybody home? inferring activity from smart home network traffic," in *Proceedings of the 2016 IEEE Security and Privacy Workshops (SPW)*, pp. 245–251, San Jose, CA, USA, May 2016.

[15] S. M. Kennedy, *Encrypted traffic analysis on smart speakers with deep learning*, Ph.D. thesis, University of Cincinnati, Cincinnati, OH, USA, 2019.

[16] V. F. Taylor, R. Spolaor, M. Conti, and I. Martinovic, "Robust smartphone app identification via encrypted network traffic analysis," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 1, pp. 63–78, 2018.

[17] M. Yair, M. Bohadana, A. Shabtai et al., "Profiliot: a machine learning approach for iot device identification based on network traffic analysis," in *Proceedings of the Symposium on Applied Computing, SAC '17*, pp. 506–509, Association for Computing Machinery, Marrakesh, Morocco, April 2017.

[18] M. Skowron, A. Janicki, and W. Mazurczyk, "Traffic fingerprinting attacks on internet of things using machine learning," *IEEE Access*, vol. 8, Article ID 20386, 2020.

[19] H. Li, Z. Xu, H. Zhu, D. Ma, S. Li, and K. Xing, "Demographics inference through wi-fi network traffic analysis," in *Proceedings of the IEEE INFOCOM 2016—The 35th Annual IEEE International Conference on Computer Communications*, pp. 1–9, San Francisco, CA, USA, April 2016.

[20] S. Feghhi and D. J. Leith, "A web traffic analysis attack using only timing information," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 8, pp. 1747–1759, 2016.

[21] M. Yang, X. Gu, Z. Ling, C. Yin, and J. Luo, "An active deanonymizing attack against tor web traffic," *Tsinghua Science and Technology*, vol. 22, no. 6, pp. 702–713, 2017.

[22] T. G. Ejeta and H. J. Kim, "Website fingerprinting attack on psiphon and its forensic analysis," in *Proceedings of the International Workshop on Digital Watermarking*, pp. 42–51, Springer, Magdeburg, Germany, August 2017.

[23] L. Basyoni, N. Fetais, A. Erbad, A. Mohamed, and M. Guizani, "Traffic analysis attacks on tor: a survey," in *Proceedings of the 2020 IEEE International Conference on Informatics, IoT, and Enabling Technologies (ICIoT)*, pp. 183–188, Doha, Qatar, February 2020.

[24] R. Attarian, L. Abdi, and S. Hashemi, "Adawfpa: adaptive online website fingerprinting attack for tor anonymous network: a stream-wise paradigm," *Computer Communications*, vol. 148, pp. 74–85, 2019.

[25] K. Abe and S. Goto, "Fingerprinting attack on tor anonymity using deep learning," *Proceedings of the Asia-Pacific Advanced Network*, vol. 42, pp. 15–20, 2016.

[26] A. Iacovazzi, D. Frassinelli, and Y. Elovici, "The DUSTER attack: Tor onion service attribution based on flow watermarking with track hiding," in *Proceedings of the 22nd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2019)*, pp. 213–225, USENIX Association, Beijing, China, September 2019.

[27] J. V. Monaco, "Feasibility of a keystroke timing attack on search engines with autocomplete," in *Proceedings of the 2019 IEEE Security and Privacy Workshops (SPW)*, pp. 212–217, San Francisco, CA, USA, May 2019.

[28] N. Baroutis and M. Younis, "A novel traffic analysis attack model and base-station anonymity metrics for wireless sensor networks," *Security and Communication Networks*, vol. 9, no. 18, pp. 5892–5907, 2016.

[29] N. Msadek, R. Soua, and T. Engel, "Iot device fingerprinting: machine learning based encrypted traffic analysis," in *Proceedings of the 2019 IEEE Wireless Communications and Networking Conference (WCNC)*, Marrakech, Morocco, April 2019.

[30] V. F. Taylor, R. Spolaor, M. Conti, and I. Martinovic, "Robust smartphone app identification via encrypted network traffic analysis," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 1, pp. 63–78, 2017.

[31] F. Kausar, S. Aljumah, S. Alzaydi, and R. Alroba, "Traffic analysis attack for identifying users' online activities," *IT Professional*, vol. 21, no. 2, pp. 50–57, 2019.

[32] M. Juarez, M. Imani, M. Perry, C. Diaz, and M. Wright, "Toward an efficient website fingerprinting defense," in *Proceedings of the European Symposium on Research in Computer Security*, pp. 27–46, Springer, Heraklion, Greece, September 2016.

[33] T. Wang and I. Goldberg, "Walkie-talkie: an efficient defense against passive website fingerprinting attacks," in *Proceedings of the 26th USENIX Security Symposium (USENIX Security 17)*, pp. 1375–1390, USENIX Association, Vancouver, Canada, August 2017.

[34] C. Chen, D. E. Asoni, A. Perrig, D. Barrera, G. Danezis, and C. Troncoso, "Taranet: traffic-analysis resistant anonymity at the network layer," in *Proceedings of the 2018 IEEE European Symposium on Security and Privacy (EuroSP)*, pp. 137–152, London, UK, April 2018.

[35] J. Liu, C. Zhang, and Y. Fang, "Epic: a differential privacy framework to defend smart homes against internet traffic analysis," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 1206–1217, 2018.