Side-Channel Information Leakage of Encrypted Video Stream in Video Surveillance Systems

Hong Li[1,3], Yunhua He[2,3], Limin Sun[1,*], Xiuzhen Cheng[3] and Jiguo Yu[4]

[1] Beijing Key Laboratory of IOT Information Security Technology, Institute of Information Engineering, CAS, China
[2]School of Computer Science, Xidian University, China
[3]Department of Computer Science, The George Washington University, USA
[4]School of Information Science and Engineering, Qufu Normal University, China
{lihong, sunlimin}@iie.ac.cn, {yunhua, cheng}@gwu.edu, jiguoyu@sina.com, *corresponding author

***Abstract*—Video surveillance has been widely adopted to ensure home security in recent years. Most video encoding standards such as H.264 and MPEG-4 compress the temporal redundancy in a video stream using difference coding, which only encodes the residual image between a frame and its reference frame. Difference coding can efficiently compress a video stream, but it causes side-channel information leakage even though the video stream is encrypted, as reported in this paper. Particularly, we observe that the traffic patterns of an encrypted video stream are different when a user conducts different basic activities of daily living, which must be kept private from third parties as obliged by HIPAA regulations. We also observe that by exploiting this side-channel information leakage, attackers can readily infer a user's basic activities of daily living based on only the traffic size data of an encrypted video stream. We validate such an attack using two off-the-shelf cameras, and the results indicate that the user's basic activities of daily living can be recognized with a high accuracy.**

I. Introduction

Video surveillance systems have been widely employed to ensure home security over the last several years. Users can set up cameras in their houses to check whether someone is making off with their valuables, or what their babies or pets are doing while they are not around. Whenever a camera is powered on, it sends a live encrypted video stream to a cloud server. Users can access the cloud server to get the live video stream by their computers or smartphones. Commercial providers of live video surveillance services include Google, Samsung, and D-Link. The video surveillance market is expected to reach $42.81 billion by 2019, growing at a compound annual growth rate (CAGR) of 19.1% [1].

To save transmission bandwidth and storage space, a video encoder removes the temporal and spatial redundancy in a video stream. Temporal redundancy is caused by the high correlation or similarity between adjacent frames, while spatial redundancy is resulted from the high correlation between pixels within one frame. Most video compression standards such as MPEG-4 and H.264 reduce temporal redundancy between adjacent frames by *difference coding*, where one frame is compared with a reference frame and only pixels that have changed with respect to the reference frame are coded. Spatial redundancy is removed by using transform techniques such as discrete cosine transform.

Difference coding can compress a video stream efficiently; however, our findings reported in this paper reveal that it brings significant side-channel information leakage. Particularly, we demonstrate that an attacker can use the traffic size data of an encrypted video stream to infer a user's activities of daily living (ADLs) including dressing, self-feeding, mobility, and personal hygiene. The key intuition behind these privacy leakage is that when a user is not at home, the difference between adjacent frames is very small, resulting in a small video traffic size; but when the user is at home, the motion of the user makes the difference between adjacent frames larger, causing the traffic size to increase dramatically. Likewise, the traffic patterns are different when a user performs different activities. Large-scale and complex movements generally yield heavy and unstable traffic caused by the large and varied differences between adjacent frames. ADLs are typically very personal, especially in medical science where they are used to measure the functional status of a person [2]; thus ADLs must be kept secret from third parties as obliged by HIPAA regulations [3].

Side-channel information leakage of encrypted traffic has been extensively studied for a decade in the context of cryptographic protocols [4], [5], web applications [6], radiation signals [7], [8], [9], and encrypted VoIP streams [10]; but it has never been addressed in video surveillance. This paper reports our exploratory investigation on the side-channel information leakage problem of encrypted video surveillance traffic data. We find that the statistical properties of a video stream traffic are totally different when a user performs different activities in front of the camera. Exploring this side-channel information leakage, we propose a novel approach to infer a user's ADLs based on only the time-series traffic size data of an encrypted video stream. Our approach mainly consists of two components: *activity segmentation* and *activity recognition*. Activity segmentation splits the time-series traffic size data apart with each segment containing only one activity. Then we extract features from each segment, and use both supervised and unsupervised learning to recognize the underlying activity

978-1-4673-9953-1/16/$31.00 ©2016 IEEE

in the next step. Our real-world experiment based evaluation using two off-the-shelf commercial cameras shows that the activities of daily living can be accurately recognized. The main contributions of this paper are summarized as follows:

- This paper is the first one to investigate the side-channel information leakage problem caused by difference coding in video surveillance systems.
- We propose a novel approach to infer a user's basic activities of daily living, which are private, by exploiting the side-channel information leaked out.
- We use two off-the-shelf cameras to validate our approach and investigate the impact of various factors on the accuracy of recognizing basic activities of daily living.

The remainder of the paper is structured as follows. Section II outlines the related work. In Section III, we introduce difference coding and investigate the side-channel information leakage problem. Our activity recognition scheme is detailed in Section IV, followed by a comprehensive experimental study for performance validation in Section V. The paper is concluded with a future research discussion in Section VI.

II. Related Work

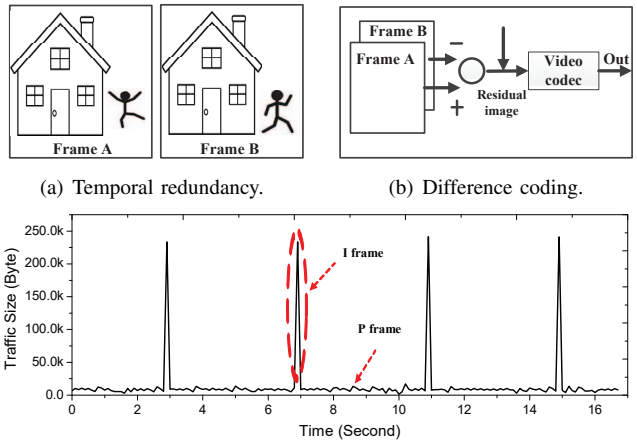
Side-channel attacks have been extensively studied in different contexts including cryptographic protocols, network applications, radiation signals, and encrypted multimedia.

In the context of cryptographic protocols: Brumley *et al.* [4] showed that time attacks could be used to extract private keys from an OpenSSL-based web server running on a machine in the local network; Song *et al.* [5] used the timing information collected from the network to learn significant information about passwords and other texts typed in SSH sessions; and Meyer *et al.* [11] presented four new Bleichenbacher side channel attacks to break SSL/TLS implementations.

In the context of network applications: Chen *et al.* [6] found that an adversary could utilize the traffic distinctions of different webpages to infer users' sensitive data in web applications even though the network traffic is encrypted; Wang *et al.* [12] proposed a scheme to identify a user's web activity by leveraging the multi-modal property of web pages; and Zhang *et al.* [13] built a system based on network traffic analysis to track a user's online activities such as chatting and downloading.

In the context of radiation signals: Xu *et al.* [7] introduced an attack that exploits the emanations of changes in light to reveal the TV programs people are watching; Srinivasan *et al.* [8] showed that an adversary could observe private activities in a home by eavesdropping on the wireless transmissions of sensors installed at the home, even when all of the transmissions are encrypted; and Zhu *et al.* [9] utilized the acoustic emanations from keystrokes to recover typed contents.

In the context of multimedia: Wright *et al.* [14] discovered that an adversary could examine patterns in the output of VoIP coder to discern considerable information about the underlying spoken language, and showed that a profile Hidden Markov Model (HMM) trained with speaker-independent and phrase-independent data could detect the presence of some phrases within encrypted VoIP calls with recall and precision exceeding 90% [10]; and Saponas *et al.* [15] determined the movie a user is watching via Slingbox by exploiting the properties of variable bitrate encoding.



(a) Temporal redundancy. (b) Difference coding.

(c) Traffic size of an encoded video stream.

Fig. 1: Temporal redundancy and difference coding.

Our work is motivated by these studies, but is different in a number of major aspects: (1) we investigate the side-channel information leakage caused by difference coding in video surveillance, which has never been explored; existing approaches summarized above can not be applied due to the fundamental differences involved in the underlying signal processing techniques; (2) we propose an approach to accurately infer users' activities of daily living by exploiting the leaked side-channel information of the video surveillance traffic.

III. Difference Coding and Side-Channel Information Leakage

A. Difference Coding

Raw video stream typically requires a large bitrate, e.g. approximately 216 Mbits for 1 second of uncompressed standard definition video [16]; thus compression is necessary for practical storage and transmissions. Typically, a video stream exhibits strong temporal redundancy caused by the high correlation or similarity between adjacent frames in the time domain as illustrated in Figure 1(a). Most video encoding standards such as MPEG-4 and H.264 exploit difference coding to remove the temporal redundancy in a video stream. The principle of difference coding is for the video codec to encode only the residual image created by subtracting every pixel in one frame from the corresponding pixel in a reference frame, as illustrated in Figure 1(b). Besides difference coding, advanced techniques such as motion estimation and compensation are used to further compress a video stream, which are out of the scope of this paper.

After compression, the video stream is encoded into successive GOPs (Group of Pictures). Within each GOP, there exist one I frame, and several P frames and B frames, where

an I frame is a reference frame that can be independently decoded without refering to other frames, a P frame holds only the difference between the current frame and the previous reference frame, and a B frame contains the residual image which is obtained by comparing with both the preceding and the following reference frames. B frames are not generally used in video surveillance due to their latency. Figure 1(c) illustrates the traffic size of an encoded video stream. The high periodic spikes in the traffic correspond to I frames, while the rest which are much smaller in size are the traffics of P frames.

B. Side-Channel Information Leakage

Difference coding can efficiently compress temporal redundancy in a video stream, but it causes significant side-channel information leakage. To verify this claim, we set up an experiment (see Figure 2(a)) using Google's Dropcam Pro, and recorded the traffic size data of an encrypted video stream without I frames when a volunteer conducted 4 basic activities of daily living (dressing, styling hair, moving, and eating), which must be kept secret from third parties as obliged by HIPAA regulations.

1) Traffic Patterns With and Without Movements: From Figure 2(b), it is observed that the traffic size is very steady and small when there is no movement in the visual view of the camera. However, it increases dramatically when certain movements are performed in front of the camera. The key reason behind this is that when there is no movement, the difference between two successive frames is very small, resulting in a small P frame size. On the contrary, when someone appears in the monitored area, his/her movements such as walking and dressing make the differences between frames much larger; thus the sizes of the P frames increase dramatically, causing violent fluctuations in the traffic of the video stream.

2) Traffic Patterns Under Different Activities: Figure 2(c)-2(f) report the traffic sizes of the encrypted video stream when different basic activities of daily living are conducted in the visual view of the camera. One can see that the traffic sizes of the video stream differ significantly when the volunteer conducts different activities. This is because activities involving large body movements (such as dressing) result in larger changes between successive frames compared to activities only involving small body movements (such as styling hair). Moreover, if an activity is composed of periodic motions among which some are more significant and some are less, there should be a periodic variation in the traffic size data. Figure 2(f) illustrates such a scenario when a user is eating. We observe that the traffic size data exhibits an obvious periodic up-and-down trend. The durations with larger traffic sizes are caused by the motions of getting food and putting it into the mouth, which are stronger body movements, while those of smaller traffic sizes are caused by chewing the food, which is only a facial movement.

Summary: These results indicate that the traffic size of an encrypted video stream without movements is much smaller than that with movements. Moreover, the traffic patterns differ significantly when a user performs different activities in the visual view of the camera, which is caused by difference coding. Such side-channel information can be exploited by an attacker to infer the private basic activities of daily living of a user, as demonstrated in the following sections.

IV. Detection of Activities of Daily Living

In this section, we propose an approach to infer a user's basic activities of daily living based on only the traffic size data of an encrypted video stream by exploiting the leaked side-channel information. Note that we focus on the case when only one person appears in the visual view of the camera, and defer the multi-person case to our future work.

A. Workflow of the Approach

As illustrated in Figure 3, the workflow of the proposed approach consists of three steps.

- *Capturing Time-Series Traffic Size Data:* At the first step we need to capture the time-series traffic size data of a video stream by sniffing the communications between a camera and the cloud server. This can be easily obtained as most commercial cameras are equipped with WiFi interfaces. Denote by $X = (t_1, t_2, t_3, ..., t_n)$ the time-series traffic size data, where t_i is the traffic size in the ith time slot Δt, and Δt is empirically set to 0.1 second.
- *Activity Segmentation:* In the second step, the time-series traffic size data is first processed to filter out the traffic sizes of all the I frames and some random noises. Then we detect the beginning point and ending point of each activity. Next we segment the time-series traffic size data X into small pieces $\{X_i\}_{i=1}^{m}$, with each segment X_i corresponding to the traffic size data of one activity.
- *Activity Recognition:* In the last step, we extract features from each segment in both time domain and frequency domain, and use machine learning algorithms to recognize the underlying activities.

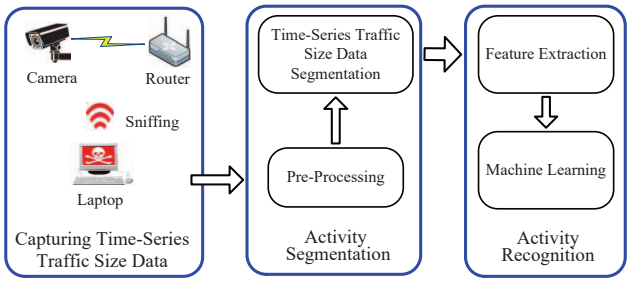
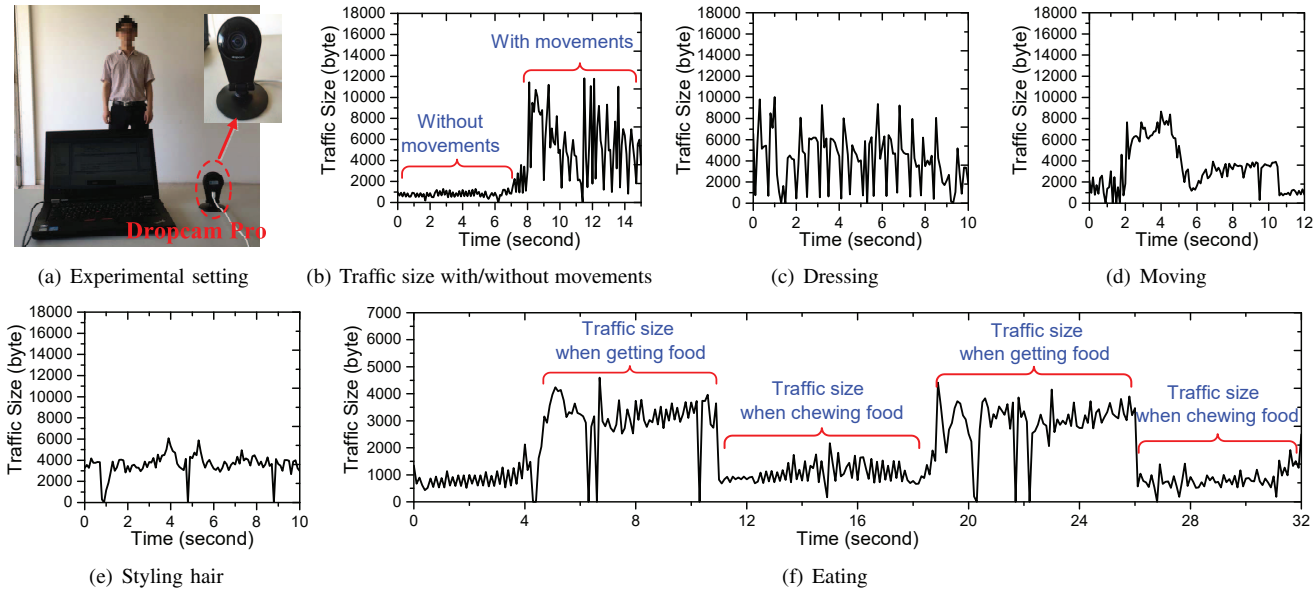


Fig. 3: Workflow of the approach

B. Activity Segmentation

1) Pre-Processing: As described in Section III-A, a user's privacy can be inferred by exploiting the side-channel information leaked out by difference coding that involves only P frames; thus we first filter out all the I frames from X. Note that I frames appear periodically in the video stream and the size of an I frame is generally much larger than that of a P



(a) Experimental setting (b) Traffic size with/without movements (c) Dressing (d) Moving

(e) Styling hair (f) Eating

Fig. 2: Traffic size of an encrypted video stream when a user performs different basic activities of daily living (all I frames are removed).

frame even when an activity occurs in the visual view of the camera, thus one can easily filter out I frames by removing the periodical high spikes in the time-series traffic size data X. To remove random noises caused by network interrupts and other factors, we employ a moving average filter.

2) Time-Series Traffic Size Data Segmentation: Assume that there are m activities within $X = (t_1, t_2, t_3, ..., t_n)$. To recognize each activity, we first locate the beginning point and ending point of each activity in X, and then split X into small segments with each corresponding to one activity. Our technique is based on the following observation: there should be a change point in the time-series traffic size data when an activity begins or ends, since the statistical properties of the traffic change. As shown in Figure 4, when dressing ends and moving starts, the traffic size rises and then drops to a lower level in a short interval. The root cause lies in that when people finish dressing and start to move, their motions change substantially, which leads to a large difference between the current frame (moving) and the reference frame (dressing). When both the current frame and the reference frame contain the motion of moving, the traffic drops to a lower level. In the following we detail the steps to locate the beginning point and ending point of each activity, and segment the time-series traffic size data.

Detecting All the Change Points: In this work, we identify the change points in the time-series traffic size data X by employing Binary Segmentation [17], which is the most widely used change point search method for one dimensional time-series data in the literature. Assume there are m change points with their positions $\tau = \{\tau_1, \tau_2, \cdots, \tau_m\}$ in X. We define $\tau_0 = 0$ and $\tau_{m+1} = n$. Consequently, the m change points split the data into $m+1$ segments $(X_{\tau_1}, ..., X_{\tau_i}, ..., X_{\tau_{m+1}})$, where

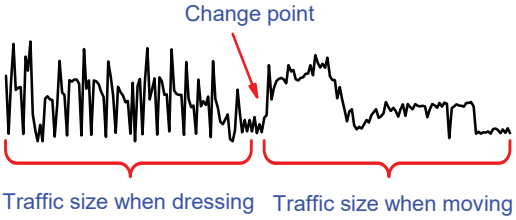


Fig. 4: The beginning point and end point of an activity

$X_{\tau_i} = \{t_{(\tau_{i-1}+1)}, ..., t_{\tau_i}\}$. Binary Segmentation identifies the change points where the statistical properties change within the time-series data by minimizing the following equation:

$$\sum_{i=1}^{m+1} [C(X_{\tau_i})] + \beta f(m) \tag{1}$$

Here C is a fitness function for a segment and $\beta f(m)$ is a penalty to guard against overfitting. In this work, we use twice the negative log likelihood as the fitness function [18], [19], denoted as follows:

$$C(X_{\tau_i}) = L(X_{\tau_i}) \cdot \{\log(2\pi) + \log[L(X_{\tau_i}) \cdot Var(X_{\tau_i})] + 1\} \tag{2}$$

where $L(X_{\tau_i})$ is the length of X_{τ_i} and $Var(X_{\tau_i})$ is the variance of the random variable X_{τ_i}. We choose $\beta = 2$ and $f(m) = \log(m)$ for penalty in this work, which are commonly used settings in literature [20].

Merging Short Segments: After getting the positions of all the change points $\tau = \{\tau_1, \tau_2, \cdots, \tau_m\}$, we split X into $(X_{\tau_1}, ..., X_{\tau_i}, ..., X_{\tau_{m+1}})$. However the above method could produce certain short segments especially at the beginning and ending of each activity. These short segments should be just

parts of certain activities as a complete activity cannot be finished within a short duration. We therefor adopt a minimum interval ε within which at least one activity of daily living can be accomplished to merge short segments. If the duration of X_{τ_i} is shorter than that threshold, i.e. $T(X_{\tau_i}) < \varepsilon$, we merge X_{τ_i} with $X_{\tau_{i+1}}$. Since moving is an activity that can last long or short, we set ε to be 4 seconds within which only moving can be accomplished.

Merging Periodic Segments: Some activity may consist of long and periodic motions, but the above method may partition it into different partial activities. For example, eating involves periodically getting food and chewing food as shown in Figure 2(f). The traffic size of getting food is much larger than that of chewing food, thus the whole activity is split into many small segments representing either getting food or chewing food. For simplicity, we treat each round of getting food and chewing food as a complete activity in this paper. In order to merge periodic motions into a complete activity, we propose the following merging algorithm.

Assume that the segments we get after the first two steps are $\{S_1, S_2, ..., S_{m'}\}$. For each S_i, we first temporarily merge its neighboring 2ω segments to get $S'_i = \{S_i, ..., S_{i+\omega-1}\}$ and $S'_{i+\omega} = \{S_{i+\omega}, ..., S_{i+2\omega-1}\}$. If the underlying activity is made up of ω periodic motions, S'_i and $S'_{i+\omega}$ should be similar. Thus we merge $\{S_i, ..., S_{i+\omega-1}\}$ if

$$dis(S'_i, S'_{i+\omega}) < \alpha \quad \text{and} \quad \left|T(S'_i) - T(S'_{i+\omega})\right| < \beta \tag{3}$$

where $dis(S'_i, S'_{i+\omega})$ is the distance between S'_i and S'_{i+1}, α and β are two thresholds, and $T(S'_i)$ is the duration of segment S'_i. Since S'_i and S'_{i+1} are two segments that vary in length, we employ dynamic time warping (DTW) [21] to calculate the distance. The algorithm of merging periodic segments is provided in Algorithm 1. We set $\omega = 2$ since only eating is periodic among the basic activities of daily living and it is made up of 2 motions. More complex activities with periodic motions are envisioned to be identified with a larger ω.

Algorithm 1 Merging Periodic Segments

Require:
Data segments, $S_1, S_2, ..., S_{num-1}$;
Threshold in time: α
Threshold in distance: β
Length of the verifying window: ω
Ensure: Activity segments;

```
Set i = 0, j = 0, periodic = False;
while i ≤ (num − 2ω + 1) do
    S'_i = {S_i, ..., S_{i+ω−1}};
    S'_{i+1} = {S_{i+ω}, ..., S_{i+2ω−1}};
    if |T(S'_i) − T(S'_{i+1})| < α & dis(S'_i, S'_{i+1}) < β then
        A_{j++} = {S_i, ..., S_{i+ω−1}};
        periodical = True;
        i = i + ω;
    else
        if periodical then
            A_{j++} = {S_i, ..., S_{i+ω−1}};
            i = i + ω;
            periodical = False;
        else
            A_{j++} = S_{i++};
        end if
    end if
end while
return A_1, A_2, ..., A_m;
```

C. Activity Recognition

After segmentation, we first filter out the segments without activities. Then we extract features from each segment and use machine learning algorithms to recognize the underlying activity.

1) Filtering out Segments without Activities: Typically there should be a segment without activity between two activities. As described in Section III, the traffic size of the encrypted video stream when there is no activity is much smaller than that when certain activities are performed in front of the camera. Thus we employ a threshold δ to filter out the segments without activities. If the average of the traffic size data in segment S is smaller than δ, S is regarded as a segment without activity. The value of δ depends on many factors such as the definition of the camera. In this work, we set δ to be 2000 bytes which is an empirical value we get through experiments.

2) Feature Extraction: If a segment is deemed to contain an activity, we need to recognize it. For this purpose we first extract features in both time and frequency domain from each segment. The features we used in this work include the *mean*, *variance*, *skewness*, and *kurtosis* of the traffic, the *duration* of the activity, and the *first k Discrete Fourier Transform (DFT) coefficients* of the segment. A detailed description of each feature is described as follows:

- *The Mean of the traffic:* This feature captures the intensity of an activity. Given a segment $X_i = (t_1, t_2, t_3, ..., t_n)$, the mean $\bar{t}$ is computed by

$$\bar{t} = \frac{1}{n}\sum_{i=1}^{n} t_i \tag{4}$$

- *The Variance of the traffic:* This feature reflects the complexity of an activity. Given a segment $X_i = (t_1, t_2, t_3, ..., t_n)$, the variance var is computed by

$$var = \frac{1}{n}\sum_{i=1}^{n} \left(t_i - \bar{t}\right)^2 \tag{5}$$

- *The Skewness of the traffic:* Skewness is a measure of symmetry of a probability distribution. Given a segment $X_i = (t_1, t_2, t_3, ..., t_n)$, the skewness of X_i can be

computed by (6), where $\bar{t}$ is the mean of the traffic.

$$skewness = \frac{\frac{1}{n}\sum_{i=1}^{n}\left(t_i - \bar{t}\right)^3}{\left(\frac{1}{n-1}\sum_{i=1}^{n}\left(t_i - \bar{t}\right)^2\right)^{3/2}} \quad (6)$$

- *Kurtosis of the traffic:* Kurtosis is a measure of whether the data are peaked or flat relative to a normal distribution. Given a segment $X_i = (t_1, t_2, t_3, ..., t_n)$, the kurtosis of X_i can be computed by (7), where $\bar{t}$ is the mean of the traffic.

$$kurtosis = \frac{\frac{1}{n}\sum_{i=1}^{n}\left(t_i - \bar{t}\right)^4}{\left(\frac{1}{n}\sum_{i=1}^{n}\left(t_i - \bar{t}\right)^2\right)^2} - 3 \quad (7)$$

- *The Duration of the activity:* This feature denotes the time duration of an activity. Given a segment $X_i = (t_1, t_2, t_3, ..., t_n)$, the duration of the segment is $n \cdot \Delta t$, where Δt is the duration of each time slot.
- *The First k DFT coefficients:* We apply DFT with a window size w to each segment and use the first k coefficients as the features in the frequency domain. From our experiments, we find that $w = 256$ and $k = 15$ perform reasonably well.

3) Machine Learning Algorithms: After the features are extracted from each segment, we employ machine learning algorithms to recognize the underlying activity of daily living. If we can get certain video streams as the training data, we adopt k-NN classification to recognize the activities; otherwise we employ DBSCAN-based clustering to perform the recognition. The two alternative approaches are described below:

k-NN Classification: The k-NN algorithm does not make any assumption about the underlying data distribution, which is useful to analyze realworld data with complex underlying distributions. A segment is first converted to a feature vector as described in Section IV-C2, and then classified by taking a majority vote of its k closest training records in the feature space. In this work, we empirically set $k = 3$.

DBSCAN-Based Clustering: We employ DBSCAN [22] as the unsupervised learning algorithm in this work since it does not need to specify the number of clusters and can find arbitrarily shaped clusters. DBSCAN is a density-based clustering algorithm. Given a set of points in some space, it groups together points that are closely packed together, and marks as outliers that lie alone in low-density regions. DBSCAN has two input parameters: $epsilon$ and $min\_samples$. We will discuss the settings of these two parameters in Section V.

4) Error Correction for the Recognition of Eating: The accuracy of segmentation can largely affect the accuracy of activity recognition. Since eating consists of periodic motions, we should perform an extra periodic segment merging while doing the activity segmentation, which could naturally cause more errors compared with other activities. To alleviate the impact of activity segmentation on the recognition accuracy of eating, we propose an error correction method. Since we treat each round of getting food and chewing food as a complete activity (i.e. eating) in the merging process, in general having a meal should involve consecutive such activities. Given $2k+1$ consecutive activities $A_{i-k}, A_{i-k+1}, ..., A_i, ...A_{i+k-1}, A_{i+k}$, if activity A_i is recognized as an outlier stating a "not eating" activity, while the majority of its $2k$ nearest neighbors in the temporal domain are identified as eating, A_i is corrected as eating. In this work, we set $k = 2$ which is an empirical value we obtained through experiments.

V. Evaluation

A. Hardware

To test our activity recognition approach, we setup a hardware testbed that includes two commercial cameras from different manufacturers, a ThinkPad T430 laptop with the Window 7 operating system, and a wireless router. The cameras we use are Google's Dropcam Pro and Samsung's SmartCam HD Pro, which are the two most popular cameras supporting video stream encryption. The video stream of Google's Dropcam Pro is encrypted by TLS/SSL (AES 128 bit), and Samsung's SmartCam HD Pro uses TLS/SSL (AES 256 bit) to keep the video stream secure. The detailed configurations of the cameras are shown in Table I. In the experiments, all the cameras send the video stream to a router using WiFi, and the laptop with WiFi interface is used to sniff the traffic between the cameras and the router.

B. Experimental Methodology

We carried out our experimental study in a typical living room with natural lighting conditions. The experiments are conducted by 4 volunteers (2 females and 2 males), with each performing 4 activities of daily living (dressing, styling hair, moving and eating) in front of the cameras for 400 times. To test the strength of our activity inference approach, we evaluated its performance under different scenarios:

- *Different Distance to the Cameras:* to evaluate the impact of activity-camera distance on the performance of our approach, we conducted experiments when all the activities are performed at 1 meter, 2 meters, and 3 meters away from the cameras.
- *Different Illumination:* to evaluate the influence of illumination on the performance of our proposed approach, we conducted experiments at different times of sunny days: 1) in the morning; 2) in the afternoon; and 3) in the evening (with light turned on).

To test the performance of k-NN classification, we varied the size of the training set by randomly picking up the training data from the whole obtained data set and using the rest as the testing set. To evaluate the performance of DBSCAN-based clustering, we randomly chose a testing set of 280 segments (the results are similar for other testing set size). Each process is repeated 100 times for statistical confidence.

C. Accuracy of Activity Segmentation

A segment is correctly identified if both its beginning point and its ending point are within a distance of 10-time-points

TABLE I: The configurations of the cameras

Camera	Configuration							
	Manufacturer	Security	Resolution	Field of view	Frame rate	Protocal	Audio	Night vision
Dropcam Pro	Google	TLS(AES 128bit)	720p HD	130°	30 frames/sec	H.264	Disabled	enabled
SmartCam HD Pro	Samsung	SSL(AES 256bit)	1080p HD	128°	30 frames/sec	H.264	Enabled	enabled

of the true positions [19]. We mainly focus on testing the segmentation accuracy when the activities are performed at different distances to the cameras, ranging from 1 m to 3 m. In this work, we use *precision*, *recall*, and *F1-score* as the metrics to measure the accuracy of the segmentation, which can be computed using the following equations:

$$precision = \frac{TP}{TP + FP} \quad (8)$$

$$recall = \frac{TP}{TP + FN} \quad (9)$$

$$F1\text{-}score = \frac{2 \times precision \times recall}{precision + recall} \quad (10)$$

where *TP* (True Positive) is the number of segments that are correctly identified, *FP* (False Positive) and *FN* (False Negative) are the numbers of wrongly accepted and wrongly rejected cases, respectively.

Table II reports the accuracy of the activity segmentation. We observe that the accuracy of the activity segmentation decreases with the increase of the distance between the activity and the cameras. The main reason is that the statistical properties of the video stream are more distinct when an activity is conducted at a closer distance to the cameras. Since the activity segmentation is based on the changes of statistical properties, it should be more accurate if the activities are performed at a closer distance. Moreover, some activities with small-scale motions can not be detected when the distance is too large. In the following evaluation only the correctly identified segments are used to evaluate the performance of the activity recognition.

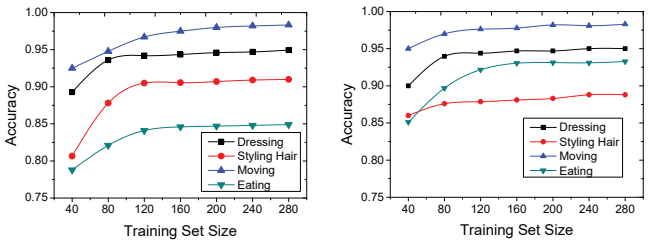
TABLE II: Accuracy of activity segmentation

Metric	Google's Dropcam Pro			Samsung's SmartCam HD Pro		
	1m	2m	3m	1m	2m	3m
Precision	94.2%	90.2%	80.4%	95.8%	95.1%	83.3%
Recall	96.5%	93.4%	84.7%	94.6%	94.3%	84.1%
F1-score	95.4%	91.8%	82.5%	95.7%	94.7%	83.7%

D. Accuracy of Classification

We varied the size of the training set to investigate its impact on the accuracy of classification. Since the testing set was randomly picked, a segment for eating may not have direct neighbors in the temporal domain; thus we did not use error correction for eating here. The results are shown in Figure 5. As expected, the accuracy of classification improves with the increase of the training set size. When the size of the training set increases from 40 to 120, the accuracies of the classification for both Google's Dropcam Pro and Samsung's SmartCam HD Pro are improved significantly. When the size of the training set goes beyond 120, the accuracy of classification does not obviously increase.

As shown in Figure 5, dressing and moving are the two activities that are the easiest to identify for both Google's Dropcam Pro and Samsung's SmartCam HD Pro. When the size of the training set goes beyond 120, the accuracy of recognizing dressing can reach 94% for both cameras, and the accuracy of recognizing moving can reach 96% for Google's Dropcam Pro and 97% for Samsung's SmartCam HD Pro. The accuracies of recognizing eating and styling hair are lower compared with those of recognizing dressing and moving. The key reason is that dressing and moving both involve larger body movements which make the traffic of the video stream exhibit more distinct features.



(a) Google's Dropcam Pro. (b) Samsung's SmartCam HD Pro.

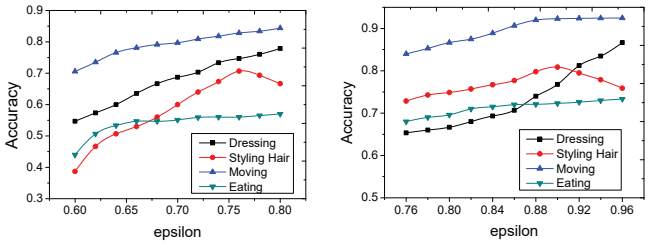
Fig. 5: Impact of training set size on classification accuracy.

E. Accuracy of Clustering

In these experiments, we tested the performance of the DBSCAN-based clustering under different settings of the parameters. For the same reason as mentioned above, we did not use error correction for eating here.

1) Impact of epsilon on the accuracy of clustering: For Google's Dropcam Pro, we set $min\_sample = 20$ and vary $epsilon$ (when min_sample is set to other values, the results are similar). In this setting when $epsilon$ is smaller than 0.6, more than 4 clusters are detected by DBSCAN. When $epsilon$ is larger than 0.8, less than 4 clusters are recognized. Only when $epsilon$ is between 0.6 and 0.8, the number of clusters is correctly identified. The accuracy of DBSCAN-based clustering for Google's Dropcam Pro is shown in Figure 6(a). It is observed that the accuracy of the clustering for each activity is significantly improved when $epsilon$ increases from 0.6 to 0.76. However, when $epsilon$ reaches 0.76, the accuracy of clustering for styling hair begins to decline. The reason behind this phenomena is that more and more activities for styling hair are wrongly recognized as dressing when $epsilon$ goes beyond 0.76, which in turn makes the recognition of dressing more accurate.

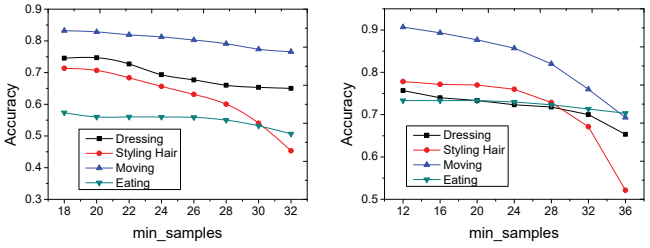
For Samsung's SmartCam HD Pro, we set $min\_sample = 15$ and vary $epsilon$ (other settings of min_sample yield very similar results). When $epsilon$ is smaller than 0.76 or larger than 0.96, the number of clusters are not correctly determined. When $epsilon$ is between 0.76 and 0.96, the accuracy of DBSCAN-based clustering is shown in Figure 6(b). We observe that the accuracy of clustering for each activity is improved when $epsilon$ increases from 0.76 to 0.9. When $epsilon$ reaches 0.9, the accuracy of clustering for styling hair begins to decline. The reason is the same as that in Dropcam Pro.



(a) Google's Dropcam Pro. (b) Samsung's SmartCam HD Pro.

Fig. 6: Impact of $epsilon$ on the clustering accuracy.

2) Impact of min_sample on the accuracy of clustering: To test the impact of min_sample on the accuracy of DBSCAN-based clustering, we set $epsilon = 0.76$ for Google's Dropcam Pro and set $epsilon = 0.96$ for Samsung's SmartCam HD Pro (Other settings of $epsilon$ yield very similar results). The impact of min_sample on the accuracy of clustering is shown in Figure 7. For Google's Dropcam Pro, when min_sample is smaller than 18 or larger than 32, the number of clusters are more or less than 4. For Samsung's SmartCam HD Pro, only when min_sample is between 12 and 36, the number of clusters can be correctly identified. It is observed that the accuracy of clustering declines with the increase of min_sample because more and more activities are regarded as outliers.

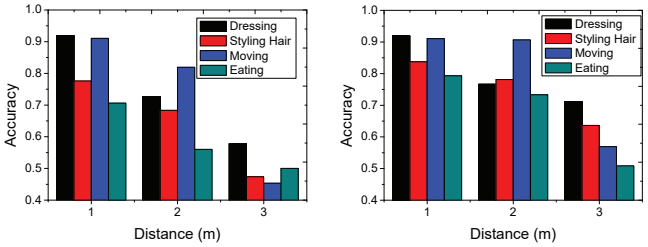


(a) Google's Dropcam Pro. (b) Samsung's SmartCam HD Pro.

Fig. 7: Impact of min_sample on the clustering accuracy.

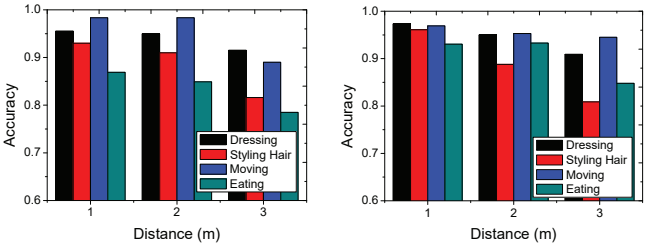
F. Varying Distance between the Cameras and Activities

In this subsection, we evaluate the impact of distance between cameras and activities on the accuracy of the activity recognition. We conducted experiments at 1m, 2m, and 3m away from the cameras. The accuracies of clustering and classification are shown in Figure 8 and Figure 9, respectively. It is observed that the accuracy of both clustering and classification declines with the increase of the distance. This is expected, as the motions occupy more pixels in a frame when the activities are performed at a closer distance to the cameras, which makes the statistical features of the underlying traffic more distinct. As indicated in Figure 8, the clustering for moving is the most susceptible to distance. When the activity is performed at 3m away from the cameras, many segments for moving are identified as outliers in the DBSACN-based clustering, which makes it become the most difficult one to recognize.



(a) Google's Dropcam Pro. (b) Samsung's SmartCam HD Pro.

Fig. 8: Impact of distance on the clustering accuracy.



(a) Google's Dropcam Pro. (b) Samsung's SmartCam HD Pro.

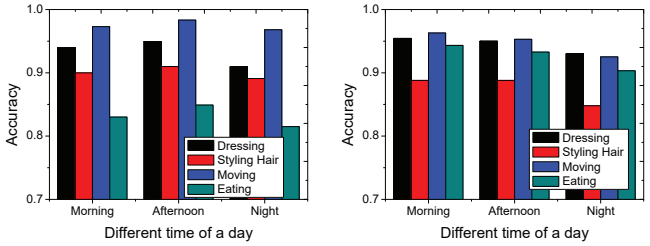
Fig. 9: Impact of distance on the classification accuracy.

G. Impact of Illumination

We further evaluate our approach in different times of a day to test the impact of illumination on the accuracy of classification and clustering. Figures 10 and 11 illustrate the accuracy of the activity recognition in different times of a sunny day. As indicated by the results, illumination affects the accuracy of both classification and clustering. The activity recognition in the afternoon is more accurate than that in the morning and in the night. The activity recognition in the night is the worst case. In the afternoon, the indoor environment is well-lit, which makes the vision of the cameras more clearer. On the contrary, the cameras can not capture the details of some motions due to the dim-light condition in the evening.

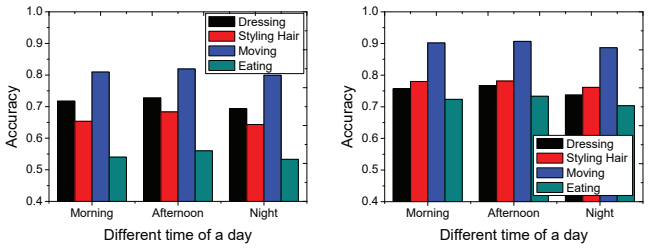
H. Impact of Error Correction on the Recognition of Eating

To evaluate the impact of error correction on the recognition of eating, we chose the data set within which the segments for eating are continuous in time to compare the recognition



(a) Google's Dropcam Pro. (b) Samsung's SmartCam HD Pro.

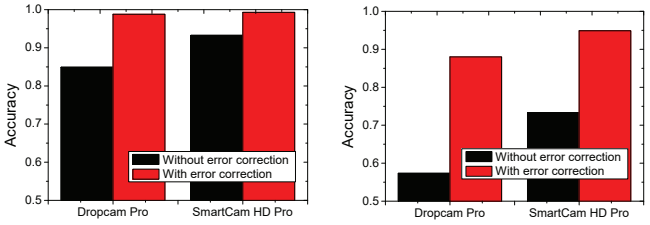
Fig. 10: Impact of illumination on the classification accuracy.



(a) Google's Dropcam Pro. (b) Samsung's SmartCam HD Pro.

Fig. 11: Impact of illumination on the clustering accuracy.

accuracy with and without error correction. As shown in Figure 12(a), the accuracies of classification for eating are only 84% and 93% for Dropcam Pro and SmartCam HD Pro, respectively. After applying error correction, the accuracies of classification reach more than 98% for both cameras. The impact of error correction on the clustering is illustrated in Figure 12(b). It is observed that the accuracy of clustering for eating improves dramatically after error correction.



(a) Accuracy of classification. (b) Accuracy of clustering.

Fig. 12: Impact of error correction.

VI. Conclusion and Future Research

In this work, we studied the side-channel information leakage problem of encrypted video streams caused by difference coding. Our findings indicate that the traffic patterns of a camera differ significantly when a user performs different activities of daily living such as dressing, styling hair, moving, and eating. By exploiting this side-channel information leakage, we developed a method to infer a user's activities of daily living based only on the size of the encrypted traffic of a video stream. Our experimental results demonstrate that one can easily recognize a user's daily activities with high accuracy.

In our future research, we will consider more complicated scenarios such as the recognition of multiple people and develop countermeasures for user's privacy protection.

Acknowledgement

This work was supported by the NSF of China (61472418 and 61373027), the Xinjiang Uygur Autonomous Region Science and Technology Project (No.201230122), the National Defense Science and Technology Innovation Fund (No.CXJJ-14-Z68), and the NSF of the US (CCF-1442642).

References

[1] ReporterLinker, "Video surveillance and vsaas market - global industry analysis, size, share, growth, trends and forecast, 2013-2019," Transparency Market Research, Tech. Rep., October 2013.

[2] J. M. Wiener, R. J. Hanley, R. Clark, and J. F. Van Nostrand, "Measuring the activities of daily living: Comparisons across national surveys," *Journal of Gerontology*, vol. 45, no. 6, pp. S229–S237, 1990.

[3] U. S. department of health and human services. Hipaa regulations and standards. [Online]. Available: http://www.hhs.gov/ocr/hipaa/

[4] D. Brumley and D. Boneh, "Remote timing attacks are practical," *Computer Networks*, vol. 48, no. 5, pp. 701–716, 2005.

[5] D. Song, D. Wagner, and X. Tian, "Timing analysis of keystrokes and ssh timing attacks," in *USENIX Security Symposium*, 2001, pp. 337–352.

[6] S. Chen, R. Wang, X. Wang, and K. Zhang, "Side-channel leaks in web applications: A reality today, a challenge tomorrow," in *IEEE Symposium on Security and Privacy*, 2010, pp. 191–206.

[7] Y. Xu, J.-M. Frahm, and F. Monrose, "Watching the watchers: Automatically inferring tv content from outdoor light effusions," in *ACM CCS*, 2014, pp. 418–428.

[8] V. Srinivasan, J. Stankovic, and K. Whitehouse, "Protecting your daily in-home activity information from a wireless snooping attack," in *ACM UbiComp*, 2008, pp. 202–211.

[9] T. Zhu, Q. Ma, S. Zhang, and Y. Liu, "Context-free attacks using keyboard acoustic emanations," in *ACM CCS*, 2014, pp. 453–464.

[10] C. V. Wright, L. Ballard, S. E. Coull, F. Monrose, and G. M. Masson, "Spot me if you can: Uncovering spoken phrases in encrypted voip conversations," in *IEEE Symposium on Security and Privacy*, 2008, pp. 35–49.

[11] C. Meyer, J. Somorovsky, E. Weiss, J. Schwenk, S. Schinzel, and E. Tews, "Revisiting ssl/tls implementations: New bleichenbacher side channels and attacks," in *USENIX Security Symposium*, 2014, p. 17.

[12] T. Wang, X. Cai, R. Nithyanand, R. Johnson, and I. Goldberg, "Effective attacks and provable defenses for website fingerprinting," in *USENIX Security Symposium*, 2014, pp. 143–157.

[13] F. Zhang, W. He, X. Liu, and P. G. Bridges, "Inferring users' online activities through traffic analysis," in *ACM WiSec*, 2011, pp. 59–70.

[14] C. V. Wright, L. Ballard, F. Monrose, and G. M. Masson, "Language identification of encrypted voip traffic: Alejandra y roberto or alice and bob?" in *USENIX Security*, 2007, pp. 43–54.

[15] T. S. Saponas, J. Lester, C. Hartung, S. Agarwal, T. Kohno *et al.*, "Devices that tell on you: Privacy trends in consumer ubiquitous computing." in *Usenix Security Symposium*, 2007, pp. 55–70.

[16] I. E. Richardson, *The H. 264 advanced video compression standard*. John Wiley & Sons, 2011.

[17] A. Scott and M. Knott, "A cluster analysis method for grouping means in the analysis of variance," *Biometrics*, pp. 507–512, 1974.

[18] J. Chen and A. K. Gupta, *Parametric statistical change point analysis: With applications to genetics, medicine, and finance*. Springer Science & Business Media, 2011.

[19] R. Killick, P. Fearnhead, and I. Eckley, "Optimal detection of changepoints with a linear computational cost," *Journal of the American Statistical Association*, vol. 107, no. 500, pp. 1590–1598, 2012.

[20] R. Killick and I. Eckley, "changepoint: An r package for changepoint analysis," *Journal of Statistical Software*, vol. 58, no. 3, pp. 1–19, 2014.

[21] S. Salvador and P. Chan, "Toward accurate dynamic time warping in linear time and space," *Intelligent Data Analysis*, vol. 11, no. 5, pp. 561–580, 2007.

[22] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise." in *Kdd*, vol. 96, no. 34, 1996, pp. 226–231.