Rethinking Privacy Risks from Wireless Surveillance Camera

QIANYI HUANG, Sun Yat-sen University & Peng Cheng Laboratory
YOUJING LU, Shanghai Jiao Tong University
ZHICHENG LUO, Southern University of Science and Technology
HAO WANG, Rutgers University
FAN WU and GUIHAI CHEN, Shanghai Jiao Tong University
QIAN ZHANG, Hong Kong University of Science and Technology

Wireless home surveillance cameras are gaining popularity in elderly/baby care and burglary detection in an effort to make our life safer than ever before. However, even though the camera's traffic is encrypted, attackers can still infer what the residents are doing at home. Although this security loophole has been reported, it does not attract much attention from the public, as it requires the attacker to be in close proximity to the camera and have some prior knowledge about the victims. Due to these requirements, the attacker has a low chance of success in the real world. In this article, we argue that the capability of attackers has been greatly underestimated. First, the attacker can leverage the characteristics of video transport protocols to recover the metadata of missing packets. Second, the attacker can build the inference model using the public datasets and adapt the model to the real traffic. Thus, the attacker can launch the attack at a distance from the camera, without prior knowledge about the victim. We also implement this attack scenario and verify that the attacker can infer the victims' activities at a distance as large as 40 m without any knowledge about the victim, neither personal nor environmental.

CCS Concepts: • **Security and privacy** → **Mobile and wireless security**; • **Networks** → **Mobile and wireless security;**

Additional Key Words and Phrases: Home surveillance camera, security and privacy

ACM Reference format:
Qianyi Huang, Youjing Lu, Zhicheng Luo, Hao Wang, Fan Wu, Guihai Chen, and Qian Zhang. 2023. Rethinking Privacy Risks from Wireless Surveillance Camera. *ACM Trans. Sensor Netw.* 19, 3, Article 60 (March 2023), 21 pages.
https://doi.org/10.1145/3570504

This research is supported in part by the Key-Area Research and Development Program of Guangdong Province (No. 2020B0101390001), in part by the National Natural Science Foundation of China (No. 62002150, 62025204, 61972252), in part by RGC under Contract CERG 16203719, Contract 16204820, and Contract R8015, and in part by The Major Key Project of PCL Department of Broadband Communication.
Authors' addresses: Q. Huang, Sun Yat-sen University & Peng Cheng Laboratory; email: huangqy@pcl.ac.cn; Y. Lu (corresponding author), F. Wu, and G. Chen, Shanghai Jiao Tong University; emails: luyoujing@sjtu.edu.cn, {fwu, gchen}@cs.sjtu.edu.cn; Z. Luo, Southern University of Science and Technology; email: 12032194@mail.sustech.edu.cn; H. Wang, Rutgers University; email: hw488@cs.rutgers.edu; Q. Zhang, Hong Kong University of Science and Technology; email: qianzh@cse.ust.hk.

1550-4859/2023/03-ART60 $15.00
https://doi.org/10.1145/3570504

1 INTRODUCTION

The global market of home security camera was valued at 3.71 billion dollars in 2019, and this value is expected to reach 11.9 billion in 2027, growing at a compound annual rate of 15.7% [22]. The prevalence of home security camera makes it possible for remote monitoring of the elderly and infants and detecting burglaries. People believe that home security cameras are making life safer than ever before.

However, this is not the truth. These networked cameras raise privacy concerns. To save storage and network bandwidth, videos are compressed before transmission. Widely used compression techniques are **variable-bitrate (VBR)** encoding, where the bit rate of the video stream is closely related to the video content. Thus, side information can be leaked from the network trace, even when the traffic is encrypted [3, 9, 10, 13–15, 17, 18, 21, 29, 30, 33]. In Reference [14], residents' activities of daily life (e.g., dressing, styling hair, moving and eating) can be inferred from the camera's Wi-Fi traffic. Similar privacy issues are reported in Reference [15], where the traffic pattern from surveillance cameras can be used to identify the best time for physical attacks, e.g., burglary.

Although the issue has been reported in previous literature, it does not attract much attention from the public. There are two possible reasons why the public underestimates the potential risk. First, current attacks assume that the attacker is in close proximity to the victim's camera. In Reference [14, 31], the sniffer is placed next to the camera, so that the sniffer can capture almost all the Wi-Fi packets transmitted by the camera. However, to launch the attack in a clandestine way, an attacker usually sniffs the wireless traffic at a certain distance away from the surveillance camera. It results in incomplete network trace due to path loss and interference. Second, it requires massive labeled data to build the inference model. However, it is difficult, if not impossible, to collect the activity label from the targeted victim. Given these two weaknesses, the public is not very concerned about the potential privacy risks from wireless surveillance cameras.

We argue that the capability of attackers has been greatly underestimated. On the one hand, the vulnerable region is not limited to the close proximity of the victim camera. We observe that the traffic of wireless cameras shows certain patterns. Even though we only capture a portion of the wireless traffic, we can still infer the missing part leveraging the inherent relationship between successive packets. It implies that even when the attacker is far from the victim's camera and can only receive some of the transmitted packets, he/she can still recover the metadata of missing packets and measure the data rate variation of the traffic flow. On the other hand, there are massive public video datasets that can serve as the labeled training data to build the inference model. Although the attacker does not know the activity label of the victim, he/she can build a machine learning model using public datasets and generate the model to the target domain.

In this article, we assess the risk of privacy leakage from wireless surveillance cameras under real scenarios. We first evaluate the vulnerable region. We analyze over 3.7 million video streaming packets and find that, in legacy video transport protocols, there are several specific types of packets, and the transition probability/interval among them has clear statistical patterns. These statistical patterns serve as the basis for deriving the missing packets. For attackers in the distance who may miss a large portion of the traffic, we propose an algorithm to reconstruct the traffic series. Second, we present an approach to infer the victims' home activity in an unsupervised manner. We take the network traffic generated from the public, labelled videos as the prior knowledge. Then we use domain adversarial transfer learning to help transfer the knowledge from the public video datasets to the target victims. We implement this attack scenario on two commercial wireless cameras, and results show that the attacker can still successfully launch attacks even at 40 m away from the victim's camera, when he/she has no idea about the victim or the environments. It implies that an attacker on the ground floor can infer the activities of residents living on the 8th floor.

The main contributions of this work are summarized as follows:

- We recognize that privacy risks from wireless cameras are greatly underestimated. The vulnerable distance is as large as 40 m, and the attacker does not need to have prior knowledge about the victim.
- We propose an algorithm for distant attackers to recover the missing part of the traffic trace. We analyze the statistical patterns of video transport protocols and design a dynamic programming algorithm to infer the metadata of missing packets.
- We present a method to infer victims' home activity in an unsupervised manner. We take advantage of the public video datasets to generate surveillance traffic as the source domain and then use domain adversarial transfer learning to transfer the knowledge from the source domain to the target domain, i.e., the real traffic from the camera.

The rest of the article is organized as follows. Section 2 introduces the video compression mechanism that leads to the privacy leakage and discusses issues in the existing attack model. Section 3 gives our attack scenario that is closer to the actual capability of attackers. Section 4 and Section 5 illustrate the algorithm for missing packet recovery and activity recognition, respectively. Section 6 evaluates the performance of the packet recovery algorithm and activity recognition accuracy. Section 7 discusses the capability of attackers and the countermeasures to this side channel attack. Section 8 introduces the related work. Section 9 concludes this article.

2 PRIVACY LEAKAGE FROM WIRELESS CAMERAS

In this section, we first introduce the video compression mechanisms that lead to privacy leakage from wireless cameras. Then we demonstrate that the risk has not been properly assessed in exiting works.

2.1 Privacy Leakage from Encrypted Traffic

Video is composed of many pictures and adjacent frames have strong correlations and similarities, as shown in Figure 1. To improve the transmission and storage efficiency, most surveillance cameras choose to use VBR encoding (e.g., H.264 encoding) to reduce the redundancies between adjacent pictures, where the size of video stream is closely related with the video content. In H.264 encoding, the video is represented as a series of **Group of Pictures (GOPs)**, and each GOP is composed of an I frame, several P frames, and B frames. An I frame contains a complete picture, which is a reference frame that can be independently decoded without referring to other frames. A P frame records the difference between the current frame and the previous reference frame. A B frame stores the difference between the preceding frame and the following reference frame. This is the basic idea of differential coding. We can know that an I frame contains more data, as shown in the right part of Figure 1. The periodic spikes in the traffic correspond to I frames, while the rest, which are much smaller in size, correspond to P frames.

Due to the characteristics of variable-bitrate compression, although all the data are encrypted, the traffic pattern of wireless cameras can result in side channel leakage. To verify this claim, in Figure 2, we present some traffic traces (without I frames) of video streams when a user performs some activities. From these traces, we have the following two clues:

The traffic size implies static or dynamic scenes. Figure 2(a) shows the traffic size with and without movements. We can observe that the traffic is light when there is no movement in the camera's field of view. The difference between two successive frames is very small so that the traffic is light. The traffic increases dramatically when the camera captures certain movements (e.g., a man sits on the sofa watching TV for a while and then stands up to do some exercise). The reason is that the difference between two successive frames is large (the captured frames keep

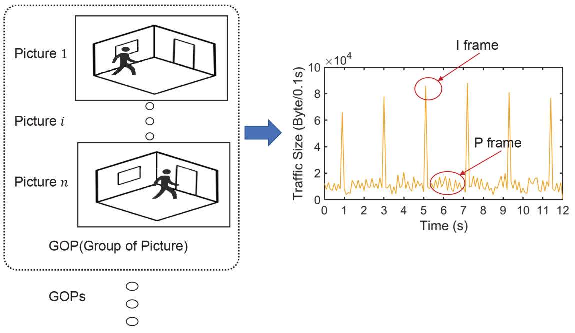
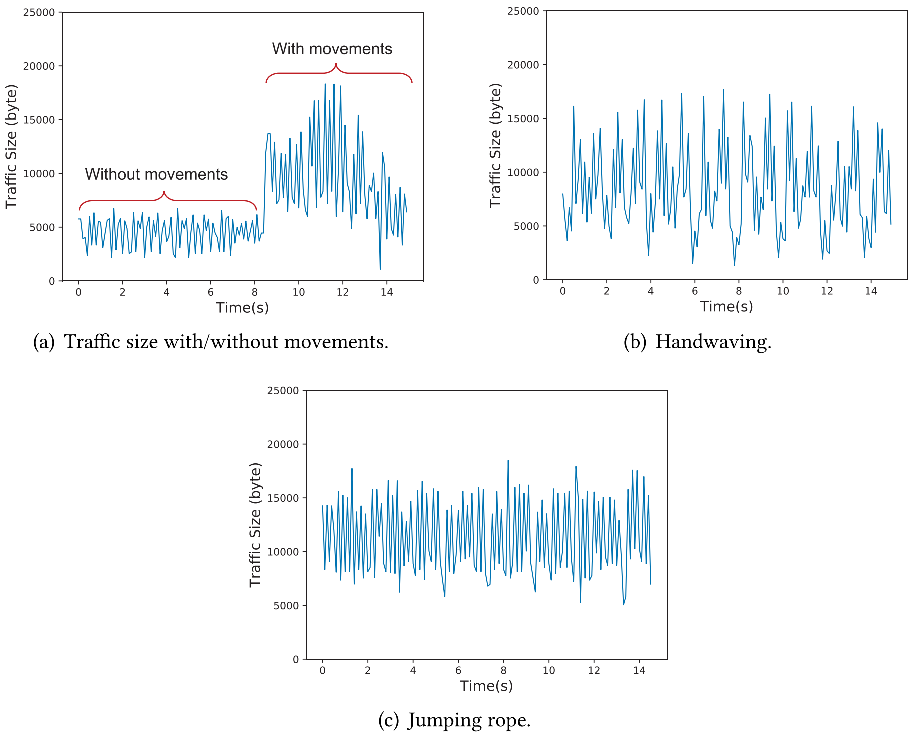


Fig. 1. H.264 encoding.



(a) Traffic size with/without movements.

(b) Handwaving.

(c) Jumping rope.

Fig. 2. Traffic size of a video stream when a user performs different types of activities (all I frames are removed).

changing when a user performs activities in front of the camera), and more differences need to be recorded, resulting in heavy traffic.

The traffic pattern implies activity type. Figure 2(b) and Figure 2(c) show the traffic pattern of handwaving and jumping rope, respectively. We can see that the traffic patterns of the two activities are significantly different. There are two key reasons: One is that the activities involving large body movements (such as jumping rope) result in larger differences between successive frames compared to activities only involving small body movements (such as handwaving); another is that the activities involving fast body movements (such as jumping rope) result in faster changes compared to activities involving slow body movements (such as handwaving).

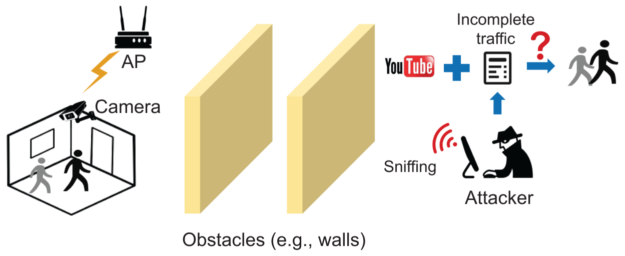


Fig. 3. Attack scenario.

2.2 Assess the Risk of Privacy Leakage

As wireless communication is essentially broadcasting, even though the data are encrypted, the volume of traffic can be acquired by other listening devices in the surrounding are. An attacker can sniff the packets transmitted by the camera and infer users' activity from the traffic volume. Existing works usually make two assumptions: (1) the attacker is in close proximity to the victim and (2) the attacker has some labels for the victim's activity. We argue that these two assumptions may not hold in reality.

2.2.1 Attackers in Close Proximity. Existing works assume that the attacker is in close proximity to the victim's camera [4, 14, 31], e.g., the sniffer is around 20 cm away from the camera in the experimental setting [14]. However, it is hard for the attacker to get so close to the victim's camera. We implement the attacker scenario in Reference [14], and the results show that when the attacker can capture 99.5% of the packets, the recognition accuracy is 0.81. When the attacker is about 15 m away and can only capture 81.20% of packets, the accuracy drops to 0.42. This shows that the attacker is unlikely to succeed when he/she is at some distance away from the victim.

2.2.2 Known Activity Labels of the Victims. Previous efforts need labels to train a classifier to infer activity from sniffed camera traffic. They assume that the attacker can collect labels of the victim's home activities. However, in real scenarios, it is unreasonable for the victims to voluntarily disclose their privacy. To test the performance when we do not have activity labels for the target domain, we take the camera traffic collected from one subject in a room as the training samples and take the camera traffic collected in another room from the same subject as the testing samples. We also use the kNN model in Reference [14], and the recognition accuracy drops to 0.41. It implies that models do not have good generality across different domains. It shows that the attacker is unlikely to succeed when he/she does not have labeled samples from the target domain.

In the following sections, we show that these two assumptions are not necessary for inferring user's activities and the public has greatly underestimated the attackers' capability. We first present the attack scenarios and then introduce how attackers can take advantage of the vulnerability to the maximum degree. The attacking process mainly contains two steps: packet recovery and activity recognition.

3 ATTACK SCENARIO

Figure 3 illustrates the attack scenario.

Camera and Wi-Fi access point: Surveillance camera monitoring user activities at home are frequently used for elder/baby monitoring and burglary detection. As the de-facto standard, cameras encode and compress videos using VBR schemes. The camera transmits packets wirelessly to the Wi-Fi AP for cloud storage or live streaming. The connection between the camera and the AP is encrypted by the standard encryption algorithms, e.g., WEP/WPA/WPA2, as defined in 802.11. The transmission power of the camera and AP follows FCC regulation, that is, 20 dBm at maximum.

Attacker: The attacker does not have the password to the victim's home WLAN, but he/she can eavesdrop on the on-going transmission between the camera and the AP. Although all data communication are encrypted, in 802.11, the MAC layer header is in plaintext, which means that the metadata of packets (e.g., source/destination address, packet length, timestamp and sequence number) is encryption free. Thus, the attacker can record the metadata from the (encryption-free) MAC layer header.

There are two steps to identify camera traffic. The first is to divide network traffic into packet flows. Although the payload is encrypted by WPA/WPA2-PSK, the MAC-layer header is encryption free. Thus, we can obtain the MAC address of the 802.11 frame and separate Wi-Fi traffic into individual packet flows. The second is to identify camera traffic from the set of packet flows. The adversary can identify Wi-Fi cameras by analyzing traffic patterns. As shown in Reference [26], the traffic patterns of cameras are significantly different from other IoT devices. The authors in Reference [26] identify two key features: (1) the ratio of sent to received data traffic and (2) the ratio of the traffic volume in bytes and in frames for a device. With these two features, we can identify cameras with high accuracy. As this is not in the scope of this study, here we simply assume that the camera traffic has been correctly identified.

The attacker is at some distance from the camera. The attacker and the camera may also be blocked by obstacles, such as several layers of concrete walls. Due to path loss and interference, the attacker cannot record all packets sent by the camera. In addition, the attacker has no prior knowledge about the residents, house/apartment layout, and the deployment position/orientation of the camera. The attacker tries to infer the private information of the victim, such as whether the house is empty, or the life patterns of the residents, such as when the residents leave home for work and when they will be back home from work. Such information leakage can lead to property loss or even life-threatening events.

We assume that the attacker has strong capabilities of learning and computation. He/she also has full access to all the public datasets. For the ease of presentation, sometimes we put ourselves in the role of the attacker to illustrate the attacking process.

4 RECOVER THE MISSING PACKETS

In this section, we present the packet recovery algorithm. We utilize the features of video transport protocols to recover the missing packets. In the first step, we analyze the statistical features of packets and the time intervals between successive packets. Then, given these statistics, we model the packet recovery as an optimization problem to find the packet sequence that has the largest likelihood. Finally, we design a dynamic programming algorithm to solve this optimization problem.

4.1 Analyze the Encrypted Video Stream

The data generated from differential coding are transmitted by video transport protocols. The transport protocols used by surveillance cameras can be mainly divided into two categories: the first is TCP-based transport protocols, e.g., Real Time Messaging Protocol and Real Time Streaming Protocol, and the second is UDP-based transport protocols, e.g., Real-time Transport Protocol and Secure Reliable Transport. For ease of presentation, we take a TCP-based protocol as an example. The situation is the similar in UDP-based protocols.

We observe the traffic from a commercial wireless camera, TP-Link TL-IPC42C-4 [28]. Figure 4 gives a snapshot of the traffic pattern when the camera is streaming live video via Wi-Fi. There are three types of packets as follows:

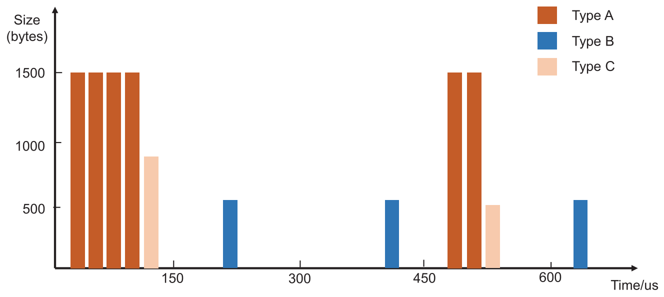


Fig. 4. Packet series captured from TP-Link TL-IPC42C-4.

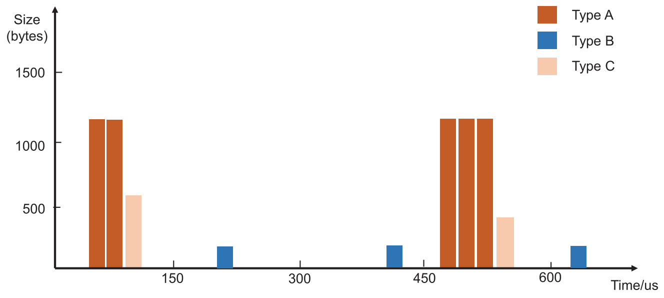


Fig. 5. Packet series captured from 360 PTZ 1080P.

(1) Type A: maximum length unit, e.g., 1,500 bytes
(2) Type B: fixed size but less than 1,500 bytes
(3) Type C: unfixed size

A sequence of type A packets appear back to back followed by a type C packet, while type B packets appear at a certain frequency (about 30 Hz). This characteristic is possibly due to the packetization of video frames. As video frames are usually larger than the maximum packet size, a GOP typically span multiple packets, i.e., a number of type A packets (full payload) and a type C packet (the remaining part). The type B packets possibly maintain the connection and carry status information of the camera, which is sent at a predefined frequency. The similar pattern is observed in another wireless surveillance camera, PTZ 1080P from 360, whose traffic pattern is shown in Figure 5. Although the sizes of type A/B packets differ, the transfer patterns are similar to Figure 4. Thus, we believe that there is a number of commercial cameras on the market that share the same features.

Based on this observation, we analyze the traffic pattern of videos from YouTube and other popular video streaming websites, covering different kinds of activities with various backgrounds. The entire video length is 558 minutes, with 3.7 million packets in total. We inspect each pair of adjacent packets and analyze the transfer probability between different types of packets. The results are shown in Table 1. The rows indicate the preceding packet and the columns indicate the present packet. We can see that there are large transfer probabilities from type A to type A, type C to type B, and small transfer probabilities from type A to type B, type B to type C, and type C to type A and C. Therefore, the transfer probabilities between different types of packets have obvious characteristics. We also analyze the time interval between different types of adjacent packets. Table 2 gives the average results. We can see that the transfer time from A to A and C is obviously shorter than that from A to B. We also estimate the average size of type C packets. These statistic values give us the opportunity to infer the type of missing packets.

Table 1. The Statistical Results of Transfer Probability

Packet	A	B	C
A	0.765	6.027e-06	0.235
B	0.437	0.543	0.020
C	0.002	0.995	0.003

Table 2. The Statistical Results of Transfer Time

Packet	A	B	C
A	0.62 (ms)	14.31 (ms)	0.52 (ms)
B	14.36 (ms)	32.29 (ms)	14.12 (ms)
C	12.91 (ms)	14.86 (ms)	13.16 (ms)

Although all the payload has been encrypted, from the plain-text packet headers, we can still obtain the metadata of the packet, including the arrival time, packet size, and the MAC layer sequence number. Suppose that the traffic captured by the attacker is $\mathbb{C} = \{p_1 = (t_1, s_1, n_1), \ldots, p_M = (t_M, s_M, n_M)\}$. A tuple (t_i, s_i, n_i) denotes the ith captured packet, where its arrival time is t_i, packet size is s_i, the sequence number is n_i, and the total number of captured packets is M. The sequence number field can reflect how many packets are missing. For example, when we receive two successive packets, one with sequence number 2563, the other one with sequence number 2567, we know that there are three packets missing between these two packets.

4.2 Problem Formulation

Suppose that there are missing packets between packet i and packet k. The number of missing packets is $(n_k - n_i - 1)$, and the total time gap is $(t_k - t_i)$. We need to find a series of packets $\mathbb{Z} = \{p_{i+1} = (t_{i+1}, s_{i+1}, n_{i+1}), \ldots, p_{k-1} = (t_{k-1}, s_{k-1}, n_{k-1})\}$, which maximizes the transfer probability and satisfies the following constraints:

$$\text{Maximize} \quad \prod_{\tau=i}^{k-1} P(p_\tau, p_{\tau+1}), \tag{1}$$

$$\text{Subject to} \quad \left| \sum_{\tau=i}^{k-1} T(p_\tau, p_{\tau+1}) - (t_k - t_i) \right| \leq \delta, \tag{2}$$

where $P(a, b)$ is the transfer probability from packet a to packet b, $T(a, b)$ is the time interval between packet a and packet b, and δ is the time error threshold. Thus, we need to find a sequence of packets that maximizes the transfer probability while their total time duration differs from the actual time gap $t_k - t_i$ within the threshold δ.

4.3 Packet Recovery Algorithm

A naive solution to this problem is to traverse all possible combinations of the missing packets and choose the combination with the largest probability. It is straightforward, but its time complexity increases exponentially with the number of missing packets (e.g., the size of searching space is 3^{50} when 50 packets are missing).

To reduce the time complexity, we design a **dynamic programming (DP)** algorithm. Since there could be any number of packets missing in the sequence, it means that we need to solve

ALGORITHM 1: DP Table Generation

```
Input: The transfer probability matrix: $P$;
       Transfer time matrix: $T$;
       Maximum number of uncaptured packets: $M$;
       The maximum time gap: $G$;
       Time error threshold: $\delta$.
Output: DP Table: $\mathbb{T}$.
1  $\mathbb{T}$ = zeros($M+1, 3, 3, G$), $Sum\_Time$ = zeros($M+1, 3, 3$);
2  foreach $\alpha$ = 0 to 2 do
3      foreach $\beta$ = 0 to 2 do
4          $Sum\_Time[0][\alpha][\beta] = T[\alpha][\beta]$;
5          $\mathbb{T}[0][\alpha][\beta][T[\alpha][\beta]] = P[\alpha][\beta]$
6  foreach $\tau$ = 1 to $M$ do
7      foreach $\beta$ = 0 to 2 do
8          foreach $\kappa$ = 0 to $G$ do
               /* Find maximum probability                                  */
9              $d$ = zeros(3);
10             $s$ = zeros(3);
11             foreach $c$ = 0 to 2 do
12                 $temp$ = zeros(3);
13                 foreach $t$ = 0 to 2 do
14                     if $T[c][\beta] - \delta \le \kappa - Sum\_Time[\tau-1][t][c] \le T[c][\beta] + \delta$ then
15                         $temp[t] = P[c][\beta] \times \mathbb{T}[\tau-1][t][c][\text{Sum\_Time}[\tau-1][t][c]]$;
16                 $t = temp.index(max(temp))$;
17                 $d[c] = temp[t]$;
18                 $s[c] = \text{Sum\_Time}[\tau-1][t][c]$
19             if $max(d) > 0$ then
20                 $\alpha = d$.index(max(d));
21                 $\mathbb{T}[\tau][\alpha][\beta][\kappa] = \max(d)$;
22                 $\text{Sum\_Time}[\tau][\alpha][\beta] = s[\alpha] + T[\alpha][\beta]$
23 return $\mathbb{T}$;
```

the similar problems for many times. Therefore, we consider generating a DP table to record the intermediate results for reference and reduce the time complexity.

Algorithm 1 outputs a four-dimensional table $\mathbb{T}$, where $\mathbb{T}[\tau][\alpha][\beta][\kappa]$ represents the maximum probability when the number of missing packets is τ, and the time gap is κ, the type of the last packet (τth) in the sequence is β (packet types A, B, C are transformed to 0, 1, 2 in the algorithms). $Sum\_Time[\tau][\alpha][\beta]$ stores the time spent for the previous τ packets when the type of the $(\tau-1)$th packet is α and the τth packet is β. α records the type of the preceding packet before the current packet β, which is determined by comparing the transfer probabilities (line 11–22 in Algorithm 1). More specifically, line 11–18 iterates over three types of packets and judges whether the time constraint can be satisfied (line 14). If the time constraint can be satisfied, then the algorithm calculates the transfer probability from the $(\tau-1)$th packet (type c) to the τth packet (type β). Line 16 and 17 determines the maximum probability when the $(\tau-1)$th packet is c and the τth packet is β. Then, the type with the maximum probability for the $(\tau-1)$th packet is chosen (line 19–22). Table $\mathbb{T}$ is used in Algorithm 2.

ALGORITHM 2: Packet Reconstruction

```
Input: Probability matrix: $P$, time matrix: $T$;
       Packets $i : (t_i, s_i, n_i)$, $k : (t_k, s_k, n_k)$;
       DP Table: $\mathbb{T}$, Time error threshold: $\delta$.
Output: Reconstructed packet series: $\mathbb{Z}$.
1  $\mathcal{M}_{\text{uncap}} = n_k - n_i - 1$, $\mathcal{G}_{\text{gap}} = t_k - t_i$, $\mathbb{Z} = [\varnothing]$;
2  foreach $\omega = \mathcal{G}_{gap} - \delta$ to $\mathcal{G}_{gap} + \delta$ do
3      $\kappa = \omega$, $\tau = \mathcal{M}_{\text{uncap}}$;
4      $t = [\varnothing]$, $\beta = s_k$, flag = 1;
       /* Search and find transfer pairs */
5      while $\tau >= 0$ do
6          $d$ = zeros(3);
7          foreach $\alpha = 0$ to 2 do
8              if $\kappa \geq 0$ and $\mathbb{T}[\tau][\alpha][\beta][\kappa] > 0$ then
9                  flag = 0; $d[\alpha] = \mathbb{T}[\tau][\alpha][\beta][\kappa]$;
10         if flag == 1 then
11             break;
12         $\alpha = d$.index(max($d$));
13         $t$.append($\beta, \alpha$);
14         $\kappa = \kappa - T[\alpha][\beta]$, $\beta = \alpha$, $\tau = \tau - 1$;
15     if flag == 1 then
16         continue;
17     $t_{\text{all}} = 0$, $t$.reverse();
18     foreach $u = 0 : 2 : t.size()$ do
19         $t_{\text{all}} = t_{\text{all}} + T[t[u]][t[u] + 1]$;
20     $t_{\text{sub}} = 0$;
21     foreach $z = 0 : 2 : t.size()$ do
22         $t_{\text{sub}} = t_{\text{sub}} + T[t[u]][t[u] + 1]$;
23         $\mathbb{Z}$.append(($t_i + \mathcal{G}_{\text{gap}} * (t_{\text{sub}} \div t_{\text{all}})$, type($t[z + 1]$).$len$, $n_i + \frac{z}{2} + 1$));
24 return $\mathbb{Z}$;
```

We give an example to help illustrate Algorithm 2, as shown in Figure 6. When we check the sequence number of sniffed packets, we find that there are four packets missing between packet 2 and packet 7. We know the type of packet 2 (B) and packet 7 (B) and also the time gap between them. Then we determine the type of packet 6, 5, 4, and 3, from back to front. To determine packet 6, we search Table $\mathbb{T}$ to find a type that satisfies the time constraint and has the maximum probability to transfer to type B (packet 7). Then we get a pair (B,C), which means the type of packet 6 is C. Next, we search Table $\mathbb{T}$ again to determine the type of packet 5, and we get (C,A), which means the type of packet 5 is A. We repeat the above operations for 5 times and get five transfer pairs. We reverse and merge them. Here we get an optimal sequence of missing packets.

Algorithm 2 shows the process of packet recovery. Given the type of current packet β, we search Table $\mathbb{T}$ and record every pair of current packet and the preceding one into a list t, as shown in line 5 to 12. The idea of Line 8 and 9 is to record the transfer probability from packet α to packet β (Line 9) when $\mathbb{T}[\tau][\alpha][\beta][\kappa] > 0$ holds (Line 8). $\mathbb{T}$ is the dynamic programing table returned by Algorithm 1. $\mathbb{T}[\tau][\alpha][\beta][\kappa]$ stores the probability when there are τ missing packets, the time duration of the missing packet sequence is κ, and the τth packet is α and the succeeding packet is

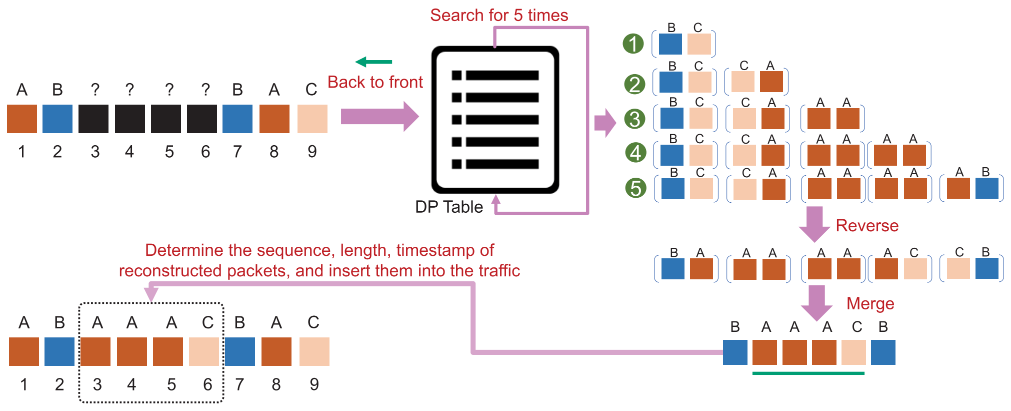


Fig. 6. An example for missing packet reconstruction.

β, respectively. We record its probability in Line 9. In Line 12–14, we store the solution with the highest probability and continue the procedure with the remaining sequence.

Then we reverse the list t, as shown in line 15 to 17. According to the list t, the time gap $\mathcal{G}_{\text{gap}}$, and the transfer time matrix, we determine the timestamp, size, and sequence number of every reconstructed packet (Lines 18 to 21). The algorithm returns the series of reconstructed packets $\mathbb{Z}$.

We next analyze the computation complexity of the proposed dynamic programming algorithm. For DP table generation, the complexity is $O(\mathcal{M} \times 3 \times 3 \times \mathcal{G}) = O(\mathcal{M}\mathcal{G})$. The complexity of algorithms increases linearly with the time resolution we used in the algorithms. When the time gap is 1s and the timeline is discretized by 10 μs, G is 1×10^5, which implies high complexity. Fortunately, Algorithm 1 only needs to be run for once. It generates the table $\mathbb{T}$, which will be used in Algorithm 2. Thus, Algorithm 1 can be run offline beforehand.

For Algorithm 2, the complexity is $O(\mathcal{M}_{\text{uncap}} \times \delta)$, where δ is the error bound for the time gap. It is used to account for inaccuracies in time measurement due to factors such as clock drift and packet delay, which is usually a small number. Thus, the complexity of Algorithm 2 is acceptable. From Figure 10(c), we can see that the packet recovery algorithm can run in real time.

5 ACTIVITY RECOGNITION

In this section, we present how to infer user activities from camera traffic in an unsupervised manner. Although the attacker has no prior knowledge about the victim and the background layout/furniture, he/she can use public video datasets to build the recognition model. Then, with some observed (unlabeled) traffic from the victim, the attacker can apply domain adaptation [7] to transfer the model from the source domain (e.g., the video traffic generated by public datasets) to the target domain (e.g., the traffic captured by the attacker). We first introduce the data pre-processing step and then the domain adaptation part.

5.1 Pre-processing

As we mentioned in Section 2.1, I frames contain complete pictures, which serves as reference frames, and P frames record the difference between the current frame and the previous reference frame. Thus, the size of the I frame is subject to the environment in the camera's field of view. To remove the effects of surrounding environment, we remove all I frames from the traffic log. As we can see from Figure 1, the size of I frames is much larger than P frames, and thus we can set a threshold and discard traffic segments that are larger than the threshold. Then, we segment the time-series of traffic size into segments of activities using the method in Reference [14].

There are a number of public video datasets that contain various user activities at different scenarios. They can serve as the labelled training samples for activity recognition. The KTH [24] and UCF-101 [27] datasets contain six kinds of human activities (a total of 2,391 videos from four scenarios and 25 people) and 101 kinds of activities (a total of 13,320 videos from YouTube), respectively. However, these datasets are not ready for building the activity inference model. We need to transform the video files into packet sequences. To achieve this goal, we place the surveillance camera in front of a computer screen, which is playing the videos in the datasets one by one. The camera captures the frames displayed on the screen and streams the live video to a Wi-Fi AP. A sniff nearby logs the packets sent by the camera.

We use $\mathbb{C}$ to denote the traffic captured from the victim camera and use $\mathbb{D}$ to denote the traffic generated by the public datasets. Since cameras may have various frame rates and resolution, which are user defined configurations, the traffic volume also varies under different settings. To cancel out the effects of these factors, we align the two datasets by Z-score normalization, where

$$\hat{s} = \frac{s - \bar{s}}{\sigma}, \tag{3}$$

where s is the original size of the packet, $\bar{s}/\sigma$ is the mean/standard deviation of packet size in the dataset, and $\hat{s}$ is the normalized value of s. We perform the Z-score normalization on both $\mathbb{C}$ and $\mathbb{D}$.

For every segment containing an activity, we use its mean, variance, skewness, kurtosis, and the first k Discrete Fourier Transform coefficients as the input to the next-stage processing. Details about these features can be found in Reference [14].

5.2 Domain-Adversarial Transfer Learning

As we have no prior knowledge about the victim and the surrounding environment, we apply the **Domain-Adversarial Neural Networks (DANN)** [6, 7] to help recognize user activity from the sniffed traffic.

In a DANN, an input sample contains $\mathbf{x} \in X$ and certain label (output) $y \in Y$, where X is the input space and Y is the label space. Two distributions $\mathcal{S}(\mathbf{x}, y)$ and $\mathcal{T}(\mathbf{x}, y)$ on $X \otimes Y$ are the source domain and the target domain, respectively, which both are assumed complex and unknown. In our case, the traffic $\mathbb{D}$ generated from the public datasets correspond to the source domain $\mathcal{S}$. The sniffed traffic $\mathbb{C}$ from the victim is the target domain $\mathcal{T}$. We want to predict the activity label of the samples in the target domain.

We next directly use the symbols $\mathcal{S}$ and $\mathcal{T}$ to illustrate the learning process of DANN. At training stage, we have an access to a large number of training samples $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ from both source domain $\mathcal{S}$ and target domain $\mathcal{T}$. d_i is denoted as the domain label, indicating the ith sample $\mathbf{x}_i$ is from which domain ($\mathbf{x}_i \sim \mathcal{S}(\mathbf{x})$ if $d_i = 0$, or $\mathbf{x}_i \sim \mathcal{T}(\mathbf{x})$ if $d_i = 1$). For the sample from the source domain ($d_i = 0$), the corresponding label $y_i \in Y$ are known in training stage. For the sample from the target domain ($d_i = 1$), the corresponding label $y_i \in Y$ are unknown in the training stage, and we want to predict its label in the test stage.

DANN has a deep feed-forward architecture, as shown in Figure 7, and the architecture is divided into three parts. The first part is a feature extractor G_f, which maps the input $\mathbf{x}$ to a D-dimensional feature vector $\mathbf{f} \in \mathbb{R}^D$, which represents the feature extracted from the processed video traffic. The feature extractor G_f includes several feed-forward layers and we denote the parameters of all layers as θ_f, i.e., $\mathbf{f} = G_\mathrm{f}(\mathbf{x}, \theta_\mathrm{f})$. The second part is a label predictor G_y, which maps the feature vector $\mathbf{f}$ to the label y, and we denote the parameters in the label predictor as θ_y. The last part is a domain classifier $G_\mathbf{d}$, which maps the feature vector $\mathbf{f}$ to the domain label d, and the corresponding parameters are θ_d.

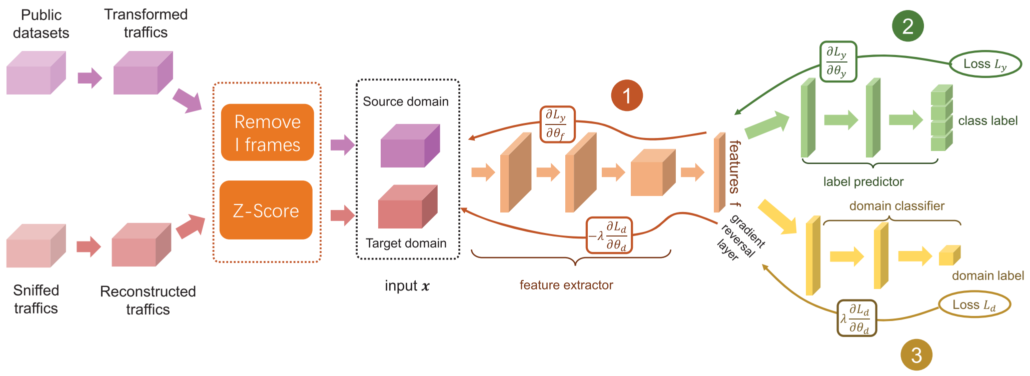


Fig. 7. The architecture of the side channel attack.

In the training stage, the first goal of DANN is to minimize the label prediction loss on the source domain part of the training set, and thus the parameters of both the feature extractor and the label predictor are optimized to minimize the loss for the source domain samples. Furthermore, the features $\mathbf{f}$ is expected to be domain invariant. Thus, the distributions $\mathcal{S}(\mathbf{f}) = \{G_\mathrm{f}(\mathbf{x};\theta_\mathrm{f})|\mathbf{x} \sim \mathcal{S}(\mathbf{x})\}$ and $\mathcal{T}(\mathbf{f}) = \{G_\mathrm{f}(\mathbf{x};\theta_\mathrm{f})|\mathbf{x} \sim \mathcal{T}(\mathbf{x})\}$ are expected to be similar.

To achieve the two goals, DANN seeks the parameters θ_f of feature extractor that maximize the loss of the domain classifier (by making the two feature distributions indistinguishable), the parameters θ_d of the domain classifier that minimize the loss of the domain classifier, while minimizing the loss of the label predictor. Therefore, the loss function is defined as follows:

$$\begin{aligned} E(\theta_\mathrm{f},\theta_\mathrm{y},\theta_\mathrm{d}) = & \sum_{i=1\cdots N, d_i=0} L_y(G_y(G_\mathrm{f}(\mathbf{x}_i;\theta_\mathrm{f});\theta_y);y_i) \\ & -\lambda \sum_{i=1\cdots N} L_d(G_d(G_\mathrm{f}(\mathbf{x}_i;\theta_\mathrm{f});\theta_d);y_i) \\ = & \sum_{i=1\cdots N, d_i=0} L_y^i(\theta_\mathrm{f},\theta_y) - \lambda \sum_{i=1\cdots N} L_d^i(\theta_\mathrm{f},\theta_d), \end{aligned} \tag{4}$$

where $L_y(\cdot,\cdot)$ denotes the loss for label prediction, $L_d(\cdot,\cdot)$ denotes the loss for the domain classification, while L_y^i and L_d^i are the corresponding loss functions evaluated at the ith training example. The parameter λ controls the tradeoff between the two objectives during the training stage.

Based on the idea of DANN, the optimal parameters $\hat{\theta}_\mathrm{f}, \hat{\theta}_y, \hat{\theta}_d$ can be sought from the following:

$$(\hat{\theta}_\mathrm{f},\hat{\theta}_y) = \arg\min_{\theta_\mathrm{f},\theta_y} E(\theta_\mathrm{f},\theta_\mathrm{y},\theta_d), \tag{5}$$

$$\hat{\theta}_d = \arg\max_{\theta_d} E(\hat{\theta}_\mathrm{f},\hat{\theta}_y,\theta_d). \tag{6}$$

Stochastic Gradient Solvers can be used to search the saddle point in Equations (5) and (6) by the following stochastic updates:

$$\theta_\mathrm{f} \leftarrow \theta_\mathrm{f} - \mu\left(\frac{\partial L_y^i}{\partial \theta_\mathrm{f}} - \lambda\frac{\partial L_d^i}{\partial \theta_\mathrm{f}}\right), \tag{7}$$

$$\theta_y \leftarrow \theta_y - \mu\frac{\partial L_y^i}{\partial \theta_y}, \tag{8}$$

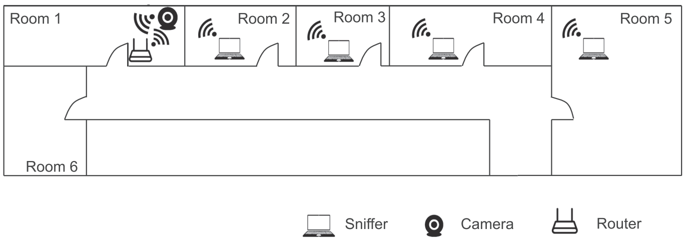


Fig. 8. Attacker sniffs the video traffic behind different number of walls.

$$\theta_d \ \leftarrow \ \theta_d - \mu \frac{\partial L_d^i}{\partial \theta_d}, \tag{9}$$

where i is a random integer in $[1, N]$, μ is the learning rate, and $-\lambda$ maximizes the domain classification loss to make the source domain and target domain indistinguishable.

6 EVALUATION

6.1 Experimental Settings

To evaluate the performance of this attack scenario, we setup a hardware testbed that includes commercial cameras, a wireless router (TP-Link TL-WR841N), a sniffer (MacBook Pro 14,1 with Intel Core i7 and 16 GB memory), and a laptop (DELL XPS 13 9360). The camera is connected to the wireless router via 802.11n, encrypted by WPA/WPA2-PSK (AES 256 bit). In our evaluation, we consider two popular wireless surveillance cameras from two manufacturers, one is TL-IPC42C-4 from TP-Link and the other is PTZ 1080P from 360. The transmission power of the wireless router is 20 dBm, which is the power limit set by FCC. Given this power, the transmission range can be as large as 100 m in open environments. The DELL laptop is connected to the same router by LAN, running a Web application to request live video streaming from the camera. It is running Wireshark to log the received traffic, which serve as the baseline. The MacBook Pro, the sniffer, is configured in the monitor mode to log the Wi-Fi packets in the air.

We conduct experiments in a conference room (named room A) and a large laboratory (named room B). We invite four volunteers to participate in the experiments. In both rooms, each volunteer performs five activities in front of the camera. The five activities include walking, jogging, running, doing jumping jacks, and jumping rope. Each activity is repeated for 10 times, and we get 400 video clips in total. We select videos corresponding to the five activities in the KTH and UCF-101 datasets as the source domain. "Walking," "jogging," and "running" are from the KTH dataset, and they account for 50% of the KTH dataset. "Jumping jacks" and "jumping rope" are from the UCF-101 dataset, and they account for 1.98% of the UCF-101 dataset. In addition, "walking," "jogging," "running," "jumping jacks," and "jumping rope" contribute 22.7%, 22.7%, 22.7%, 15.9%, and 15.9% of the training set, respectively.

To evaluate the effects of distance and obstacles on the completeness of surveillance traffic, we also conduct experiments in two different scenarios. In the first scenario, we study the effects of obstacles (e.g., walls, shelves). As shown in Figure 8, the camera and the router are placed in Room 1, while the attacker (sniffer) eavesdrops the video traffic in Rooms 2, 3, 4, and 5. In the second scenario, we study the effects of distance. As shown in Figure 9, the camera and the router are placed in Room 6, while the attacker eavesdrops the video traffic at distances of 5, 10, 20, 30, 35, and 40 m from the camera, respectively.

To evaluate the performance, we consider the following two metrics:

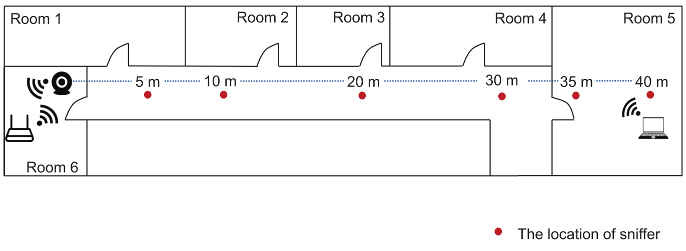


Fig. 9. Attacker sniffs the video traffic at different distance.

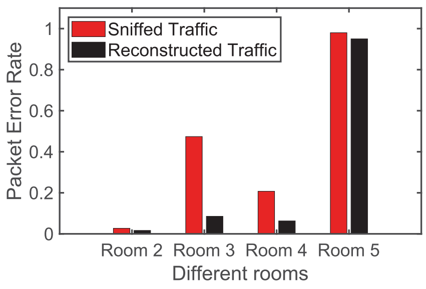
- Packet error rate: Packet error rate is defined as the ratio of the number of incorrectly received data packets over the total number of packets. A packet is declared incorrect if it is missing or it is not reconstructed correctly. For a missing packet (t_x, s_x, n_x), if the reconstructed packet $\hat{x} : (t_{\hat{x}}, s_{\hat{x}}, n_{\hat{x}})$ satisfies $s_{\hat{x}} = s_x$, $n_{\hat{x}} = n_x$, and $|t_{\hat{x}} - t_x| < \zeta$, where ζ is a threshold of time reconstruction error, then we declare that the packet is reconstructed correctly.
- Recognition accuracy: Recognition accuracy is a ratio of correctly predicted observations (activity label) to the total observations.

6.2 The Performance of Packet Reconstruction Algorithm

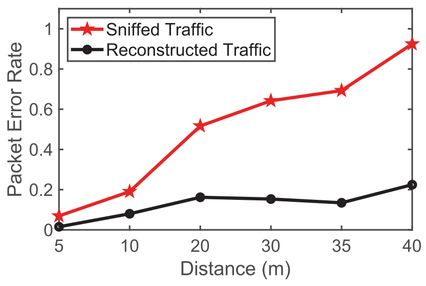
Figures 10(a) and 10(b) show the packet error rate of the sniffed and reconstructed video traffic in the two scenarios. The red bars in Figure 10(a) show the packet error rate of the video traffic sniffed in Rooms 2, 3, 4, and 5. We can see that the packet error rates of the video traffic sniffed in Rooms 3, 4, and 5 (with more walls) are higher than that sniffed in Room 2 (with only one wall). We can see that more walls and obstacles lead to higher packet error rate. The packet error rate of Room 3 is higher than that of Room 4 due to the complicated environment in Room 3. Room 3 is a server room, which is full of computers and shelves with routers and switches. The complicated multipath environment is the possible reason for high packet error rate. The black bars in Figure 10(a) show the video traffic reconstructed by the proposed algorithm. We can see the reconstructed video traffic have lower packet error rate than the original sniffed video traffic. In Room 3, the original error rate is 0.47, and the error rate is reduced to 0.085 by the packet recovery algorithm. Thus, the packet reconstructed algorithm effectively reconstruct the missing packets. However, in Room 5, the original error rate is too high (0.98), and the sniffer cannot pick up sufficient information for packet recovery.

The red line in Figure 10(b) shows the packet error rate of the video traffic sniffed at distances of 5, 10, 20, 30, 35, and 40 m. We can see the packet error rate increases with the growth of the distance. The black line in Figure 10(b) shows the video traffic reconstructed by the proposed algorithm. The packet error rate of the reconstructed video traffic is obviously lower than that of the original sniffed video traffic. When the packet error rate is high (around 90% in at 40 m), the proposed algorithm can still reconstruct the missing packets effectively. Overall, the proposed reconstruction algorithm reduces the packet error rate by 28.40% on average in our experiments.

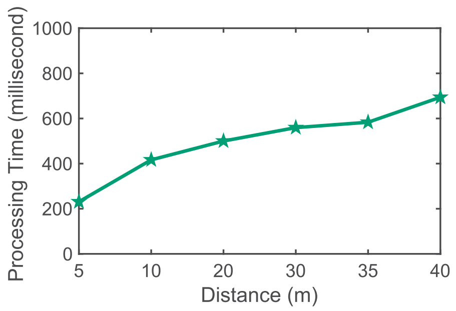
Figure 10(c) shows the processing time of packet recovery algorithm. The algorithm is running on a server with Intel i9-9900K and 64 GB memory. The processing time is defined as the runtime for recovering one second of sniffed video traffic, which is calculated by dividing the total runtime by the video length. We can see that for all cases, the processing time is lower than 1 second. It increases with the growth of distance, because it needs more time to reconstruct the incomplete traffic with higher error packet rates. Nevertheless, we can see that the attacker can reconstruct



(a) The first scenario.

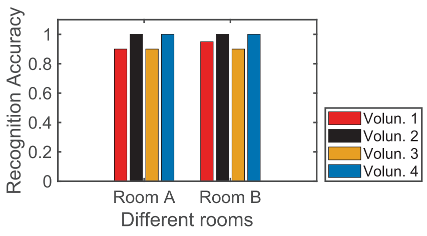


(b) The second scenario.

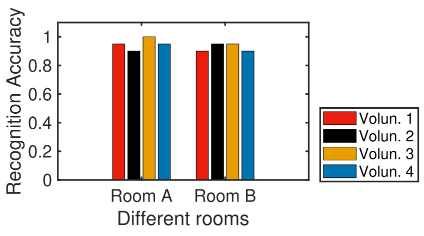


(c) The time communing of packet reconstruction.

Fig. 10. The packet error rate of sniffed and reconstructed video traffic in the two scenarios, and the running time of the packet recovery algorithm for processing 1 second of video traffic.



(a) Video traffic sniffed from TP-Link TL-IPC42C-4.



(b) Video traffic sniffed from 360 PTZ 1080P.

Fig. 11. The recognition accuracy of video traffic sniffed from the two cameras.

1 second of sniffed traffic in less than 1 second. In other words, the attacker can reconstruct the sniffed traffic in real time.

6.3 The Performance of Activity Recognition

We first evaluate the performance of activity recognition model with complete video traffic. The traffic is captured by putting the sniffer next to the camera. We use the data from public video datasets and 80% of the target subject (without activity label) to train the domain classifier and test the remaining 20% data from the target. The unlabeled data from the target can be obtained by sniffing the victim camera for a period of time. Figure 11 shows the recognition accuracy on each volunteer with video traffic sniffed from the two cameras. We can see that the average recognition accuracy is higher than 0.95. It means that the domain adversarial learning model achieves high

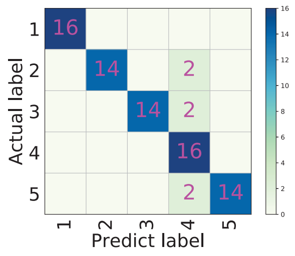


Fig. 12. The confusion matrix of activity recognition.

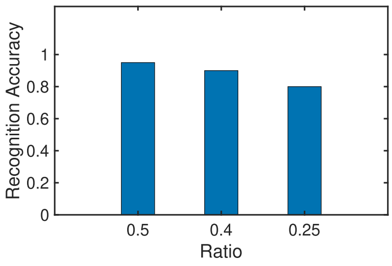


Fig. 13. The recognition accuracy under different user-to-camera distances.

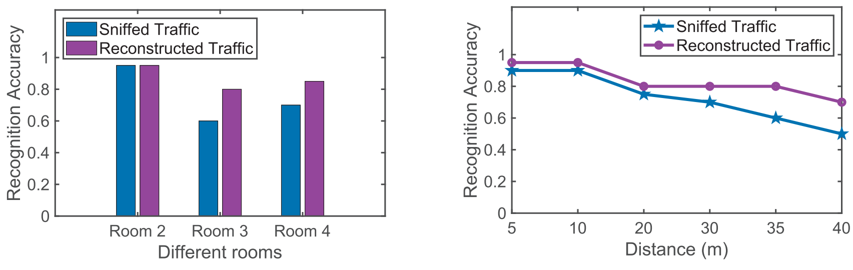
accuracy when there is no activity label for the target. Figures 11(a) and 11(b) show that there is no obvious difference of recognition accuracy between TP-Link TL-IPC42C-4 and 360 PTZ 1080P. It also indicates that the activity recognition model we learned can be applied for different camera models, as wireless cameras' traffic has similar statistical features.

Figure 12 shows the confusion matrix. Jumping jacks, jumping rope, jogging, running, and walking are labeled as "1," "2," "3," "4," and "5," respectively. For every kind of activities, there are 16 test samples in the test set. We can see that all samples of jumping jack ("1") and walking ("5") are classified correctly. For other three activities, most of their test samples are also classified correctly.

We also study the effect of user-to-camera distance on recognition accuracy. To make the experiments controllable, we place the surveillance camera in front of a computer screen, which is playing the videos involving user activities. We change the distance between the camera and the computer screen to emulate the change of user-to-camera distance. We use the ratio of Region of Interest (i.e., pixels corresponding to users) in the whole frame as the distance metric. As shown in Figure 13, the recognition accuracy decreases slowly with a decrease of the ratio (the increase of the distance). The reason is that the change of traffic pattern is less significant when the ratio of dynamic pixels in the frame decrease, and some tiny traffic pattern changes are not captured by the feature extractor.

Figure 14 shows the recognition accuracy of the sniffed and reconstructed video traffic in the two scenarios. The blue bars in Figure 14(a) show the recognition accuracy of the sniffed video traffic in the first scenario. We can see that the recognition accuracy is higher than 0.6 even when the packet error rate is higher than 0.4 (Room 3). It shows the strong capability of the transfer learning model. The purple bars in Figure 14(a) show that the reconstructed video traffic can greatly improve the accuracy of activity recognition. After packet recovery, the recognition accuracy in Room 3 improves to 0.8.

The blue line in Figure 14(b) shows the recognition accuracy of the sniffed video traffic in the second scenario. We can see that the recognition accuracy decreases with the growth of the distance (i.e., the growth of packet error rate). The reconstructed video traffic (purple line) has



(a) The first scenario. (b) The second scenario.

Fig. 14. The recognition accuracy of sniffed and reconstructed video traffic in the two scenarios.

higher recognition accuracy than the original sniffed traffic. In the worst case, where the packet error rate of the sniffed traffic is around 0.9 at a distance of 40 m, it can still achieve a recognition accuracy higher than 0.6.

We compare our results with the most relevant paper [14]. In Reference [14], the experiments were conducted by four volunteers (two females and two males), with each performing four types of daily activities (dressing, styling hair, moving, and eating) in front of the cameras for 400 times. The sniffer is next to the camera. The accuracy ranges from about 78% to 95% when the training set size is 40. In our study, from Figure 14(b), we can see that when the attacker is in close proximity of the victim (within 5 m), the recognition accuracy is about 95%, which is comparable to Reference [14], while our conditions are more challenging (the attacker has no labeled data about the victim). From above results, we can see that our study has comparable performance with Reference [14] when the attacker is in close proximity, while can handle more challenging conditions. This has not been investigated in previous study.

7 DISCUSSION

Vulnerable activities: In this article, we build inference models for five types of activities. The main obstacle to extend to a larger activity set is due to the available activities in public video datasets. In the KTH and UCF-101 datasets, a large portion of videos are unsuitable to emulate the traffic of home surveillance cameras. For example, some videos are shot by moving cameras, some are shot outdoor, and some contain activities unlikely to be captured by a surveillance camera (e.g., a woman is putting on makeup in front of the camera, where the user-to-camera distance is shorter than expected). However, the attacker's recognition capability is not limited to these five types. He/she can acquire video datasets containing the activity he/she is interested in and build the model in a similar way. In our future work, we want to explore the maximum set of activities that the attacker can recognize.

Here, we choose five types of activities from the datasets. In addition to activity recognition, the model can also give additional information. For example, the moving speed can be used to infer the age of the target, e.g., children, adults, and the elderly have different speed patterns. By combining some domain knowledge, the attacker can obtain more information. For example, the elderly usually gets up early in the morning and adults' schedule have a clear weekday-weekend pattern. This information can be used to build a detailed profile for the family.

Vulnerable cameras: There are two cases where the methods proposed in this article may fail. First, some cameras use proprietary transmission protocols such that their packets can hardly be captured by commercial off the shelf sniffers (e.g., laptops). If the adversary targets at these cameras, then he/she may need to develop corresponding packet logging tools. Second, some cameras may support local storage. The recorded videos are first stored locally and sent to the cloud server if triggered by some certain events (upon request or detecting motion). The algorithm proposed in this article may not work under this case. We leave it to our future work to study the privacy issue in this scenario.

Possible countermeasure: Some countermeasures such as traffic shaping and tunneling have been proposed [1, 19, 20, 32, 34, 35]. The main idea is to add dummy packets or change the inter-arrival time to cover up the real traffic. We argue that these strategies are not practical. First, we analyze the aggregated traffic rate but not the size of each individual packet. Note that we cannot change the inter-arrival time at random; otherwise, the video cannot play smoothly. We neither can add a lot of dummy packets to camouflage the real traffic, which will introduce extra traffic overhead and violate the initial design goal of variable-bitrate encoding. Thus, even though we may change the size/inter-arrival time for some packets, the aggregated traffic rate is still closely related to the transformation of scene. Second, it needs to design some transportation protocols to realize traffic shaping and tunneling, which can hardly be applied on COTS cameras.

Here, we also propose a possible solution that is compatible with existing cameras. A large number of security cameras support rotating to avoid blind spots. Thus, these cameras can be configured to rotate at some random time points. The change of field-of-view can cause fluctuations in the traffic volume, which can help to hide the residents' activity. For example, it is hard to tell whether the bitrate variation is caused by user entering/leaving the field or the rotation of the camera. Randomness can be exaggerated by rotating at a random timestamp, with random amplitude and speed.

8 RELATED WORK

Data Leakage from Home Surveillance Camera: Many incidents about data leakage from home surveillance camera have been reported. Wyze, a home camera manufacturer, reported the leakage of 2.4 million customer data [8]. The log-in email and credentials of 3,672 Amazon Ring cameras are also compromised, enabling intruders to access live camera footage and 30-day or 60-day video history [12]. A user of Mi Home Security Camera, provided by Xiaomi Inc., accidentally accesses images of other unknown homes [11]. These security issues mainly result from vulnerable security mechanism from the service providers or inappropriate passwords set by the users.

Privacy Leakage from Encrypted Traffic: Although network traffic is protected by cryptographic tools, researchers demonstrate that sensitive information can be inferred from traffic streams [2, 4, 5, 16, 25, 31]. Slingbox Pro's encrypted, VBR-encoded multimedia streaming leaks information about which movie the user is watching [23]. Similarly, Schuster et al. [25] used the encrypted video streams to recognize which video on YouTube/Netflix the user is browsing. Li et al. [14] exploited KNN classifier and DBSCAN-Based clustering methods to recognize the encrypted camera traffic if it contains one of the four types of user activities (i.e., dressing, styling hair, moving, and eating). Cheng et al. [4] inferred whether the members of family are at home by analyzing the bitrate variation of the wireless surveillance camera traffic. Wang et al. [31] implemented eight classifiers on the camera traffic to classify and identify fine-grained user behaviors, such as watching TV, opening/closing doors, sweeping the floor, and so on. Li et al. [15] analyzed the logs provided by a mainstream home security camera service provider in China, covering 15.4 million video streams from 211k active users. Results reveal that the video uploading patterns can reveal the users' daily routines. Apthorpe et al. [2] observed that the network traffic rates of four types of IoT devices (a sleep monitor, a web camera, a WeMo switch, and an Amazon Echo) can reveal potentially sensitive user interactions. Li et al. [16] inferred real-time communications between pairwise users and the social connections of users by analyzing the traffic data.

These works usually assume that the attacker is in close proximity of the victim, either connected to the same LAN or close enough to sniff the wireless packets. It also requires the attack to acquire the victims' activity label for building the inference model. Due to these requirements, the attacker has a low chance of success. In this article, we argue that the capability of attackers has been greatly underestimated.

9 CONCLUSION

In this article, we argue that the capabilities of attackers are greatly underestimated. We have verified that an attacker can infer victims' activities at a distance as large as 40 m without any activity label from the victims. It indicates that the attacker does not need to stay close to the victim and does not need to have any information about the victim, neither personal nor environmental. The experiment results show that the proposed dynamic programming algorithm can reduce the packet error rate by 28.40%, and the domain adversarial learning can achieve average 0.95 activity recognition accuracy on unseen subjects. Last, we discuss the countermeasure that the victims can take to resist against this side channel attack.

REFERENCES

[1] Noah Apthorpe, Danny Yuxing Huang, Dillon Reisman, Arvind Narayanan, and Nick Feamster. 2019. Keeping the smart home private with Smart(er) IoT traffic shaping. In *Proceedings of the Privacy Enhancing Technologies Symposium*. 128–148.

[2] Noah Apthorpe, Dillon Reisman, and Nick Feamster. 2017. A smart home is no castle: Privacy vulnerabilities of encrypted iot traffic. arXiv:1705.06805. Retrieved from https://arxiv.org/abs/1705.06805.

[3] Yushi Cheng, Xiaoyu Ji, Tianyang Lu, and Wenyuan Xu. 2019. On detecting hidden wireless cameras: A traffic pattern-based approach. *IEEE Trans. Mobile Comput.* 19, 4 (2019), 907–921.

[4] Yushi Cheng, Xiaoyu Ji, Xinyan Zhou, and Wenyuan Xu. 2017. HomeSpy: Inferring user presence via encrypted traffic of home surveillance camera. In *Proceedings of the IEEE International Conference on Parallel and Distributed Systems (ICPADS'17)*. 779–782.

[5] Ran Dubin, Amit Dvir, Ofir Pele, and Ofer Hadar. 2017. I know what you saw last minute—encrypted http adaptive video streaming title classification. *IEEE Trans. Inf. Forens. Secur.* 12, 12 (2017), 3039–3049.

[6] Yaroslav Ganin and Victor Lempitsky. 2015. Unsupervised domain adaptation by backpropagation. In *Proceedings of the International Conference on Machine Learning (ICML'15)*. 1180–1189.

[7] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *J. Mach. Learn. Res.* 17, 1 (2016), 2096–2030.

[8] Sandra E. Garcia. 2019. Data Breach at Wyze Labs Exposes Information of 2.4 Million Customers. Retrieved from https://www.nytimes.com/2019/12/30/business/wyze-security-camera-breach.html.

[9] Jiaxi Gu, Jiliang Wang, Zhiwen Yu, and Kele Shen. 2018. Walls have ears: Traffic-based side-channel attack in video streaming. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM'18)*. 1538–1546.

[10] Jiaxi Gu, Jiliang Wang, Zhiwen Yu, and Kele Shen. 2019. Traffic-based side-channel attack in video streaming. *IEEE/ACM Trans. Netw.* 27, 3 (2019), 972–985.

[11] Ryne Hager. 2020. Google Shuts Down Xiaomi Access to Assistant Following Nest Hub Picking up Strangers' Camera Feeds (Update: Fully Resolved). Retrieved from https://www.androidpolice.com.

[12] Caroline Haskins. 2019. A Data Leak Exposed the Personal Information of over 3,000 Ring Users. Retrieved from https://www.buzzfeednews.com/article/carolinehaskins1/data-leak-exposes-personal-data-over-3000-ring-camera-users.

[13] Yizhen Jia, Yinhao Xiao, Jiguo Yu, Xiuzhen Cheng, Zhenkai Liang, and Zhiguo Wan. 2018. A novel graph-based mechanism for identifying traffic vulnerabilities in smart home iot. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM'18)*. 1493–1501.

[14] Hong Li, Yunhua He, Limin Sun, Xiuzhen Cheng, and Jiguo Yu. 2016. Side-channel information leakage of encrypted video stream in video surveillance systems. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM'16)*. 1–9.

[15] Jinyang Li, Zhenyu Li, Gareth Tyson, and Gaogang Xie. 2020. Your privilege gives your privacy away: An analysis of a home security camera service. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM'20)*. 387–396.

[16] Ke Li, Hong Li, Hongsong Zhu, Limin Sun, and Hui Wen. 2019. Side-channel information leakage of traffic data in instant messaging. In *Proceedings of the IEEE International Performance Computing and Communications Conference (IPCCC'19)*. 1–8.

[17] Tian Liu, Ziyu Liu, Jun Huang, Rui Tan, and Zhen Tan. 2018. Detecting wireless spy cameras via stimulating and probing. In *Proceedings of the ACM International Conference on Mobile Systems, Applications, and Services (MobiSys'18)*. 243–255.

[18] Xiaolong Liu, Jibao Wang, Ying Yang, Zigang Cao, Gang Xiong, and Wei Xia. 2019. Inferring behaviors via encrypted video surveillance traffic by machine learning. In *Proceedings of the IEEE International Conference on High Performance Computing and Communications, IEEE 17th International Conference on Smart City, and IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS'19)*. 273–280.

[19] Robert Martin, John Demme, and Simha Sethumadhavan. 2012. Timewarp: Rethinking timekeeping and performance monitoring mechanisms to mitigate side-channel attacks. In *Proceedings of the IEEE International Symposium on Computer Architecture (ISCA'12)*. 118–129.

[20] Vibhor Rastogi and Suman Nath. 2010. Differentially private aggregation of distributed time-series with transformation and encryption. In *Proceedings of the ACM Special Interest Group on Management of Data Conference (SIGMOD'10)*. 735–746.

[21] Jingjing Ren, Daniel J. Dubois, David Choffnes, Anna Maria Mandalari, Roman Kolcun, and Hamed Haddadi. 2019. Information exposure from consumer iot devices: A multidimensional, network-informed measurement approach. In *Proceedings of the ACM Internet Measurement Conference (IMC'19)*. 267–279.

[22] Grand View Research. 2020. Smart Home Security Cameras Market Size, Share & Trends Analysis Report By Product (Wired, Wireless), By Application (Doorbell Camera, Indoor Camera, Outdoor Camera), By Region, and Segment Forecasts, 2020–2027. Retrieved from https://www.grandviewresearch.com/industry-analysis/smart-home-security-camera-market.

[23] T. Scott Saponas, Jonathan Lester, Carl Hartung, Sameer Agarwal, Tadayoshi Kohno, et al. 2007. Devices that tell on you: Privacy trends in consumer ubiquitous computing. In *Proceedings of the USENIX Security Symposium (USENIX Security'07)*. 55–70.

[24] Christian Schuldt, Ivan Laptev, and Barbara Caputo. 2004. Recognizing human actions: A local SVM approach. In *Proceedings of the IEEE International Conference on Pattern Recognition (ICPR'04)*, Vol. 3. 32–36.

[25] Roei Schuster, Vitaly Shmatikov, and Eran Tromer. 2017. Beauty and the burst: Remote identification of encrypted video streams. In *Proceedings of the USENIX Security Symposium (USENIX Security'17)*. 1357–1374.

[26] Rajib Ranjan Maiti Siby, Sandra and Nils Ole Tippenhauer. 2017. IoTScanner: Detecting privacy threats in IoT neighborhoods. In *Proceedings of the 3rd ACM International Workshop on IoT Privacy, Trust, and Security*.

[27] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. 2012. UCF101: A dataset of 101 human actions classes from videos in the wild. arXiv:1212.0402. Retrieved from https://arxiv.org/abs/1212.0402.

[28] TP-Link. 2020. TP-Link TL-IPC42C-4. Retrieved from https://nkchinaimport.com/tplink-tlipc42c4-twoway-voice-200w-pixel-monitoring-network-wireless-camera-hd.html.

[29] Rahmadi Trimananda, Janus Varmarken, Athina Markopoulou, and Brian Demsky. 2020. Packet-level signatures for smart home devices. *Signature* 10, 13 (2020), 54.

[30] Yinxin Wan, Kuai Xu, Guoliang Xue, and Feng Wang. 2020. IoTArgos: A multi-layer security monitoring system for internet-of-things in smart homes. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM'20)*. 874–883.

[31] Jibao Wang, Zigang Cao, Cuicui Kang, and Gang Xiong. 2019. User behavior classification in encrypted cloud camera traffic. In *Proceedings of the IEEE Global Communications Conference (GLOBECOM'19)*. 1–6.

[32] Tao Wang and Ian Goldberg. 2017. Walkie-talkie: An efficient defense against passive website fingerprinting attacks. In *Proceedings of the USENIX Security Symposium (USENIX Security'17)*. 1375–1390.

[33] Yinhao Xiao, Yizhen Jia, Chunchi Liu, Xiuzhen Cheng, Jiguo Yu, and Weifeng Lv. 2019. Edge computing security: State of the art and challenges. *Proc. IEEE* 107, 8 (2019), 1608–1631.

[34] Fan Zhang, Wenbo He, Yangyi Chen, Zhou Li, XiaoFeng Wang, Shuo Chen, and Xue Liu. 2013. Thwarting Wi-Fi side-channel analysis through traffic demultiplexing. *IEEE Trans. Wireless Commun.* 13, 1 (2013), 86–98.

[35] Xiaokuan Zhang, Jihun Hamm, Michael K. Reiter, and Yinqian Zhang. 2019. Statistical privacy for streaming traffic. In *Proceedings of the Network and Distributed System Security Symposium (NDSS'19)*.

Received 22 March 2021; revised 9 October 2022; accepted 18 October 2022